Chapitre 11

Géométrie algorithmique

La première section de ce chapitre contient un rappel de quelques notations classiques en géométrie euclidienne, ainsi que la mise en place d'outils élémentaires qui nous seront utiles dans les sections suivantes. La deuxième section est consacrée aux algorithmes de calcul de l'enveloppe convexe d'un ensemble fini de points du plan. Nous terminons par un algorithme dynamique plus sophistiqué. Quelques problèmes de localisation d'un point dans le plan divisé en régions sont étudiés dans la troisième section; on y utilise des structures de données assez complexes, en particulier les ensembles ordonnés persistants. La dernière section est consacrée aux diagrammes de Voronoï de points et de segments.

11.1 Notions préliminaires

Le développement récent de l'algorithmique géométrique est dû aux progrès de la technologie. En effet, la synthèse d'images par ordinateur était encore, il y a quelques années, fort coûteuse. Les ordinateurs actuels ont une rapidité et une puissance de calcul qui rendent l'interactivité possible en infographie. Il est inutile de souligner l'importance de cet outil comme moyen de communication entre la machine et l'utilisateur.

La géométrie joue un rôle fondamental dans un grand nombre de domaines tels que vision par ordinateur, robotique, conception assistée par ordinateur, « design » industriel, image de synthèse, etc. Cela explique l'importance algorithmique des problèmes géométriques. Grossièrement, on peut, si l'on y tient, placer une frontière entre les mathématiques et l' informatique de la façon suivante. Le mathématicien prouve l'existence d'un objet : « l'enveloppe convexe d'un ensemble fini de points du plan est un polygone convexe » , l'informaticien le calcule (si c'est possible), et cherche un algorithme efficace (s'il en existe).

Dans ce chapitre, nous présentons des algorithmes pour quelques problèmes fondamentaux de géométrie plane. Il est intéressant de noter que certains principes algo-

rithmiques classiques tels que la dichotomie se révèlent souvent particulièrement adaptés à un certain nombre de problèmes géométriques. D'autre part, de nouvelles méthodes spécifiques à la géométrie émergent. Parmi elles, on peut citer le principe de balayage de l'espace par un hyperplan. Ce principe permet de transformer un problème statique en dimension k en un problème dynamique en dimension $k-1$, et exploite le fait qu'en cours de balayage, la «situation» étudiée n'est modifiée de manière significative qu'en un nombre fini de positions de l'hyperplan et que par ailleurs deux situations consécutives sont «peu différentes»; on dispose donc d'une grande partie de l'information déjà calculée, en utilisant par exemple la structure d'ensemble ordonné persistant. Une autre approche algorithmique fréquente consiste à subdiviser l'espace en «régions» où les solutions au problème posé sont les mêmes. C'est le cas par exemple dans la recherche des voisins : on calcule le diagramme de Voronoï de l'ensemble des sites.

11.1.1 Notations

Les objets considérés appartiennent à un plan affine euclidien orienté. Parmi les bases orthonormées directes, on en privilégie une notée $(\vec{\imath}, \vec{\jmath})$, définissant deux directions appelées horizontale $(\vec{\imath})$ et verticale $(\vec{\jmath})$.

On notera :

$\overrightarrow{pq}$	le vecteur d'origine p et d'extrémité q;
$d(p, \vec{u})$	la droite passant par p, de vecteur directeur $\vec{u}$ non nul;
$\vec{d}(p, \vec{u})$	la droite orientée par $\vec{u}$ passant par p;
$d(p, q)$	la droite passant par p et q $(p \neq q)$;
$\vec{d}(p, q)$	la droite orientée de p vers q passant par p et q $(p \neq q)$;
$[p, q]$	le segment d'extrémités p et q;
$[p, \vec{u}($	la demi-droite d'origine p de vecteur directeur $\vec{u}$;
dét $(\vec{u}, \vec{v})$	le déterminant du couple de vecteurs $(\vec{u}, \vec{v})$ dans une base orthonormée directe;
$(\vec{u}, \vec{v})$	l'angle de $\vec{u}$ avec $\vec{v}$; on notera encore $(\vec{u}, \vec{v})$ lorsqu'il n'y a pas ambiguïté le secteur angulaire associé à l'angle $(\vec{u}, \vec{v})$.

Un angle$(\vec{u}, \vec{v})$ est dit *convexe* (resp. *réflexe*, *plat*) s'il est de mesure strictement inférieure à π (resp. strictement supérieure à π, égale à π).

Etant donnée une suite de points $(x_1, ..., x_n)$, on appelle *suite circulaire* et on note $((x_1, ..., x_n))$ l'ensemble des suites $(x_i, x_{i+1}, \cdots, x_n, x_1, \cdots, x_{i-1})$ conjuguées de la suite $(x_1, ..., x_n)$. Dans une suite circulaire $((x_1, ..., x_n))$, chaque élément a un *successeur*et un *prédécesseur* unique. Le successeur de x_i est x_{i+1}, avec la convention que $x_{n+1} = x_1$. De même, le prédécesseur de x_i est x_{i-1}, avec la convention que $x_0 = x_n$.

11.1.2 Lignes polygonales, polygones

Une *ligne polygonale* est une suite $L = (S_1, ..., S_n)$ de segments $S_i = [x_i,\ x_{i+1}]$ de longueur non nulle. Les segments S_i sont les *côtés* de la ligne polygonale, et les points x_i sont ses *sommets*. On convient aussi de représenter la ligne polygonale L par la suite $(x_1, ..., x_{n+1})$ de ses sommets.

La ligne polygonale L est *fermée* si $x_{n+1} = x_1$. Elle est *simple* si deux segments non consécutifs ont une intersection vide et si deux segments consécutifs ont une intersection réduite à une extrémité et ont pour support des droites distinctes. Dans le cas où la ligne polygonale est fermée, on considère que les segments S_n et S_1 sont consécutifs.

Un *polygone* (simple) est une ligne polygonale fermée (simple) ayant au moins trois côtés.

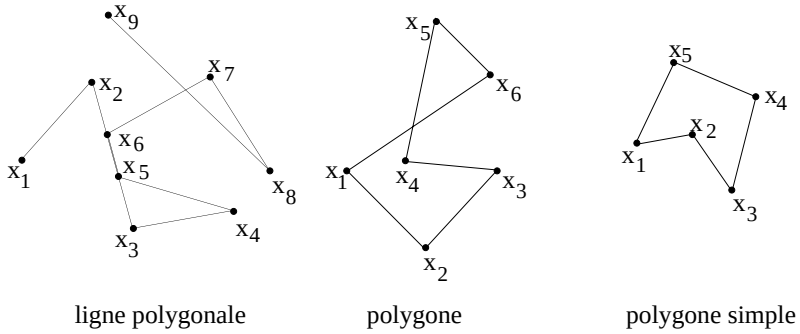

Figure 1.1: *Lignes polygonales.*

Un polygone simple détermine deux régions du plan. L'une bornée, est appelée *l'intérieur*, l'autre non bornée est appelée *l'extérieur*. La frontière de ces deux régions est l'union des côtés du polygone. Pour ne pas alourdir l'écriture, il est d'usage d'appeler encore polygone la région fermée et bornée définie par l'intérieur du polygone et sa frontière.

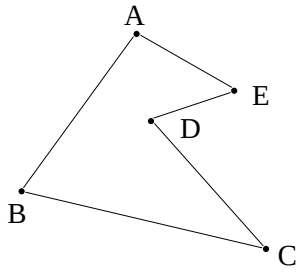

Figure 1.2: *Le contour positif du polygone est* $((A, B, C, D, E))$.

Soit $P = (x_1, ..., x_n, x_{n+1} = x_1)$ un polygone simple. Le plan étant orienté, P a un *contour positif* (ou direct) et un *contour négatif* qu'on conviendra de représenter

par les suites circulaires $((x_1, ..., x_n))$ et $((x_n, \ldots, x_1))$. Le contour positif est tel que l'intérieur du polygone se trouve à sa gauche (voir figure 1.2).

Exemple. Le contour positif du polygone P de la figure 1.2 est $((A, B, C, D, E))$ et son contour négatif est $((A, E, D, C, B))$.

11.1.3 Ordre polaire, circuit polaire

Ordre polaire

On note $m\text{mes}(\vec{u}, \vec{v})$ la mesure en radian de l'angle $(\vec{u}, \vec{v})$ dans l'intervalle $[0, 2\pi[$. Soit D une demi-droite du plan d'origine O. On définit dans le plan privé de O un ordre total appelé *ordre polaire relatif à la demi-droite* D par :

$$p \leq q$$

si et seulement si :

$$\text{mes}(D, \overrightarrow{Op}) < \text{mes}(D, \overrightarrow{Oq}) \text{ ou } (\text{mes}(D, \overrightarrow{Op}) = \text{mes}(D, \overrightarrow{Oq}) \text{ et } Oq \leq Op)$$

Autrement dit, l'ordre polaire relatif à D est l'ordre lexicographique pour les coordonnées polaires des points par rapport à D, à ceci près que l'ordre relatif à la distance à O est inversé.

Exemple. Pour les points de la figure 1.3, l'ordre polaire de S par rapport à D est $(p_6, p_5, p_4, p_2, p_1, p_3, p_7)$.

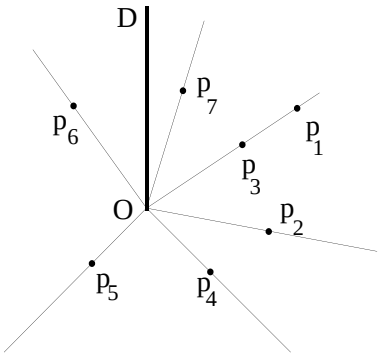

Figure 1.3: *Ordre polaire.*

Lemme 1.1. *Soit S un ensemble de points, $O \notin S$, et deux demi-droites D et D' d'origine O. Les suites obtenues en rangeant les points de S pour les deux ordres polaires relatifs à D et D' sont conjuguées l'une de l'autre.*

Ce lemme nous permet de définir une notion qui ne dépend que de S et O, celle de *circuit polaire de S relativement à O.*

Circuit polaire

Soit S un ensemble de points et $O \notin S$, on appelle *circuit polaire de S relativement à O* la suite circulaire de l'ordre polaire des points de S par rapport à une demi-droite quelconque D d'origine O. Dans l'exemple de la figure 1.3, la suite circulaire $((p_1, p_3, p_7, p_6, p_5, p_4, p_2))$ est le circuit polaire de $S = \{p_1, ..., p_7\}$ par rapport à O.

Nous introduisons ici une nouvelle notion qui permettra de calculer efficacement le circuit polaire d'un ensemble de points relativement à un point fixé. Soit O un point fixé du plan. On considère dans l'ensemble des points du plan privé de O une relation appelée *relation de précédence circulaire convexe relativement à O* notée $\preceq_O$ et définie par :

$$p \preceq_O p' \iff 0 < \text{mes}(\overrightarrow{Op}, \overrightarrow{Op'}) \leq \pi \text{ ou } (\text{mes}(\overrightarrow{Op}, \overrightarrow{Op'}) = 0 \text{ et } Op \geq Op')$$

On notera $\prec_O$ la relation stricte associée. Il est clair que $\preceq_O$ n'est pas une relation d'ordre car elle n'est ni transitive ni anti-symétrique (mais presque...). Dans l'exemple de la figure 1.4 ci-dessous, on a :

$$p_1 \prec_O p_2,\ p_2 \prec_O p_3,\ p_3 \prec_O p_1$$

Néanmoins, la relation de précédence circulaire convexe possède certaines pro-

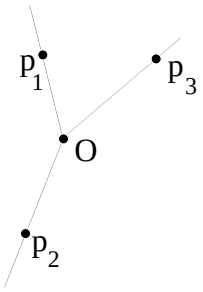

Figure 1.4: $p_1 \prec_O p_2,\ p_2 \prec_O p_3,\ p_3 \prec_O p_1$

priétés utiles, comme nous allons le voir. Notons que cette relation *se calcule en temps constant*, en effectuant uniquement des opérations élémentaires telles que multiplications, additions ou soustractions de nombres réels, sans que l'on soit obligé de calculer les mesures des angles. En effet, on a :

$$p \preceq_O p' \iff \begin{cases} \text{dét}\,(\overrightarrow{Op}, \overrightarrow{Op'}) > 0 \\ \textit{ou} \\ \text{dét}\,(\overrightarrow{Op}, \overrightarrow{Op'}) = 0,\ \overrightarrow{Op} \text{ et } \overrightarrow{Op'} \text{ de même sens, et } Op \geq Op' \\ \textit{ou} \\ \text{dét}\,(\overrightarrow{Op}, \overrightarrow{Op'}) = 0,\ \overrightarrow{Op} \text{ et } \overrightarrow{Op'} \text{ de sens contraire.} \end{cases}$$

On définit par ailleurs la relation ternaire p_1 *est entre* p_0 *et* p_2 *par rapport à* O que l'on notera $entre_O(p_0, p_1, p_2)$ par :

$$\exists i \in \{0, 1, 2\}\ \ p_i \preceq_O p_{i+1} \text{ et } p_{i+1} \preceq_O p_{i+2}$$

(les indices sont définis modulo 3.) C'est une notion étroitement liée à la notion de circuit polaire qui sera fort utile dans la suite, en effet :

Lemme 1.2. *Soit p_0, p_1, p_2 trois points distincts et distincts de O. On a $entre_O(p_0, p_1, p_2)$ si et seulement si le circuit polaire de $\{p_0, p_1, p_2\}$ relativement à O est $((p_0, p_1, p_2))$.*

On peut remarquer que dire que c est entre a et b par rapport à O signifie simplement que le point c appartient au secteur angulaire $(\overrightarrow{Oa}, \overrightarrow{Ob})$ dont une partie de la frontière est exclue, ce qui est précisé dans la figure 1.5. Les traits en gras font partie du secteur angulaire.

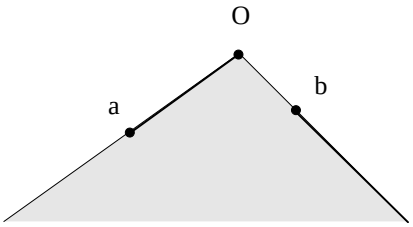

Figure 1.5: *Un secteur angulaire.*

On peut alors redéfinir les notions de circuit polaire et d'ordre polaire grâce à la relation $entre_O$:

Proposition 1.3. *La suite circulaire $((p_1, ..., p_n))$ est un circuit polaire par rapport à O si et seulement si pour tout i, le point p_i est entre p_{i-1} et p_{i+1}.*

Proposition 1.4. *Soit p_0 un point distinct de O, et soit $\leq$ l'ordre polaire par rapport à la demi-droite $[O, \overrightarrow{Op_0}($. On a l'équivalence suivante :*

$$p \leq q \iff p \text{ est entre } p_0 \text{ et } q$$

Cet énoncé fournit un algorithme de calcul du circuit polaire d'un ensemble S par rapport à un point O qui ne nécessite pas le calcul de mesures d'angles. Cela est appréciable car d'une part ces calculs sont coûteux, d'autre part ils utilisent des fonctions réciproques de fonctions trigonométriques qui engendrent des erreurs d'approximation.

Proposition 1.5. *Soit S un ensemble de N points et O un point n'appartenant pas à S. Le circuit polaire de S par rapport O se calcule en $O(NlogN)$ comparaisons entre les points pour la relation de précédence circulaire convexe.*

Preuve. Il suffit de choisir un point p_0 distinct de O, et de trier les points pour l'ordre polaire relatif à la demi-droite $[O, \overrightarrow{Op_0}($ en utilisant la propriété précédente. ∎

On peut remarquer que la relation d'ordre polaire se calcule encore plus simplement dans le cas particulier suivant :

Proposition 1.6. *Soit S un ensemble fini de points, et O un point fixé n'appartenant pas à S. Si tous les points de S appartiennent à un demi-plan ouvert dont la frontière passe par O, alors la relation de précédence circulaire convexe par rapport à O est un ordre total sur S ; c'est l'ordre polaire associé à toute demi-droite Ox non située dans ce demi-plan.*

Nous terminons par la définition de *tour gauche* ou *droit*. Soient p, q, r trois points distincts du plan. Le triplet (p, q, r) est un *tour gauche* (resp. *droit*) si le point r est situé strictement à gauche (resp. à droite) de la droite orientée $\overrightarrow{d}(p, q)$.

Proposition 1.7. *Soient p, q, r trois points non alignés du plan. Les cinq propriétés qui suivent sont équivalentes :*

- (*i*) *(p, q, r) est un tour gauche ;*
- (*ii*) *$((p, q, r))$ est le contour positif du triangle formé des trois points p, q, r ;*
- (*iii*) $\text{dét}(\overrightarrow{pq}, \overrightarrow{pr}) > 0$;
- (*iv*) *l'angle $(\overrightarrow{pq}, \overrightarrow{pr})$ est convexe non nul ;*
- (*v*) $q \prec_p r$.

Nous considérons maintenant la question de la fusion en un seul circuit de deux circuits polaires par rapport à un même point O. L'algorithme proposé est voisin de celui qu'on utilise habituellement pour la fusion de deux listes triées, mais il est adapté aux circuits qui sont de nature légèrement différente. Soient $P = ((p_0, \ldots, p_{n-1}))$, $Q = ((q_0, \ldots, q_{k-1}))$ deux circuits polaires par rapport à un point O. La suite $(p_0, \ldots, p_{n-1})$ représente la liste triée des points de P pour l'ordre polaire relativement à la demi-droite $[O, \overrightarrow{Op_0}($, de même $(q_0, \ldots, q_{k-1})$ représente la liste triée des points de Q pour l'ordre polaire relativement à la demi-droite $[O, \overrightarrow{Oq_0}($. Il suffit pour résoudre le problème de fusionner deux listes triées *pour le même ordre*, par exemple pour l'ordre polaire par rapport à $[O, \overrightarrow{Oq_0}($. Or on sait que la suite triée des points de P pour l'ordre polaire par rapport à $[O, \overrightarrow{Oq_0}($ est une suite conjuguée de la suite $(q_0, \ldots, q_{k-1})$, commençant disons par p_{i+1} (figure 1.6).

Ce point s'obtient en insérant q_0 dans le circuit P, car q_0 s'insère alors entre p_i et p_{i+1}. Il reste alors à fusionner les deux listes triées $(q_0, \ldots, q_{k-1})$ et $(p_{i+1}, \ldots, p_{n-1}, p_0, \ldots, p_i)$ pour l'ordre polaire par rapport à $[O, \overrightarrow{Oq_0}($.

Les circuits P et Q par rapport à O sont donnés sous forme de listes. L'algorithme FUSION(P, Q, O) calcule une liste qui représente le circuit fusion de P et Q par rapport au point O.

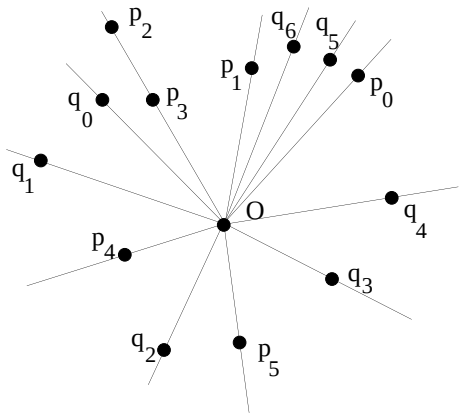

Figure 1.6: q_0 *s'insère entre* p_3 *et* p_4*, on fusionne alors* $(q_0, \ldots, q_6)$ *avec* $(p_4, p_5, p_0, \ldots, p_3)$.

Algorithme FUSION(P, Q, O);
(1) Insérer q_0 dans le circuit P; soit p_i le prédécesseur de q_0;
(2) Fusionner les listes triées $(q_0, \ldots, q_{k-1})$ et $(p_{i+1}, \ldots, p_{n-1}, p_0, \ldots, p_i)$ pour l'ordre polaire relatif à la demi-droite $[O, \overrightarrow{Oq_0}($.

L'étape (1) prend un temps $O(n)$. L'étape (2) prend un temps $O(n+k)$. Notons ici encore que la fusion des listes ne nécessite pas le calcul de l'angle polaire des points par rapport à $[O, \overrightarrow{Oq_0}($, la relation de précédence circulaire convexe suffit pour trier les points.

Proposition 1.8. *La fusion de deux circuits polaires de tailles respectives* n *et* k *se fait en temps* $O(n + k)$.

11.2 Enveloppe convexe

11.2.1 Généralités

La convexité est une propriété très intéressante en géométrie et elle intervient dans un nombre important d'algorithmes. C'est pourquoi un important travail de recherche s'est effectué sur ce thème pour s'efforcer d'obtenir des algorithmes de calcul d'enveloppe convexe qui soient performants tant en espace qu'en temps. Nous donnons ici les algorithmes les plus classiques en montrant les avantages et inconvénients de chacun, en terminant par un algorithme dynamique.

Un ensemble S est *convexe* si pour tout couple (x, y) de points de S, le segment $[x, y]$ est contenu dans S. L'*enveloppe convexe* d'un ensemble S, notée $EC(S)$, est le plus petit ensemble convexe contenant S.

Remarquons qu'une intersection quelconque d'ensembles convexes est convexe, et que le plan lui-même est convexe. Il s'ensuit que pour tout ensemble S, l'enveloppe convexe de S existe car c'est exactement l'intersection de tous les ensembles convexes contenant S. Nous nous limitons ici au calcul de l'enveloppe convexe d'un ensemble fini de points du plan. Dans ce cas, l'enveloppe convexe vérifie une propriété intéressante que l'on peut décrire par l'image suivante : on obtient l'enveloppe convexe de n points en entourant ces points avec un ruban élastique; lorsqu'on le lâche il prend la forme de l'enveloppe convexe .

Remarque. Dans toute cette section, nous ferons une légère entorse à la définition d'un polygone simple et admettrons qu'un polygone simple peut n'avoir que deux côtés, i.e. être d'intérieur vide, ceci pour donner des théorèmes aux énoncés plus simples, ainsi que des preuves sans cas particulier.

Théorème 2.1. *Soit $S \subset \mathbb{R}^2$ un ensemble de n points ($n > 1$). L'enveloppe convexe de S est un polygone convexe dont les sommets appartiennent à S.*

Avant de donner la preuve du théorème, précisons des notations que nous serons amenés à utiliser à plusieurs reprises par la suite.

Soit $((p_1, ..., p_n))$ le contour direct d'un polygone convexe P et p un point extérieur à P. Soient δ et δ' les deux demi-droites issues de p et tangentes à P, telles que l'angle (δ, δ') soit convexe. Ces demi-droites coupent le contour de P soit en un sommet de P soit en un côté de P (voir figure 2.1). Soit p_i (resp. p_j) le sommet de P le plus éloigné de p sur δ (resp. sur δ'). On dira que le secteur angulaire $(\overrightarrow{pp_i}, \overrightarrow{pp_j})$ est *le cône enveloppant P de sommet p*. Notons que p_i et p_j peuvent encore être déterminés de la façon suivante. Soit D une demi-droite ne coupant pas le polygone P, et considérons l'ordre polaire relatif à D. Alors p_i est le plus petit point pour l'ordre polaire relatif à D parmi les sommets de P; par ailleurs soit p'_j le plus grand point parmi les sommets de P, alors p_j est le sommet de P le plus loin de O sur la demi-droite $[O, \overrightarrow{Op'_j}($, c'est-à-dire le plus petit sur cette demi-droite pour la relation d'ordre que l'on considère. Les points p_i et p_j se calculent donc en temps $O(n)$ si n est le nombre de sommets de P.

Preuve. La preuve se fait par récurrence sur n. La propriété est vraie pour $n = 2$. Soit $S' \in \mathbb{R}^2$ un ensemble de n points ($n > 2$), $p \in S'$, et $S = S' - \{p\}$. Si p appartient à l'enveloppe convexe $EC(S)$ alors $EC(S') = EC(S)$. Sinon, soit $((p_1, ..., p_h))$ le contour direct de $EC(S)$, et soient p_i et p_j les sommets de $EC(S)$ tels que $(\overrightarrow{pp_i}, \overrightarrow{pp_j})$ soit le cône enveloppant $EC(S)$ de sommet p. Les sommets p_i et p_j coupent le contour direct de $EC(S)$ en deux listes telles que l'une, de p_i vers p_j, est rangée dans le sens positif pour un circuit polaire relativement à p, et l'autre, de p_j vers p_i, dans le sens négatif. En supprimant cette dernière liste, sauf les points p_i et p_j, et en la remplaçant par p on obtient le contour positif de $EC(S')$ (figure 2.2). ∎

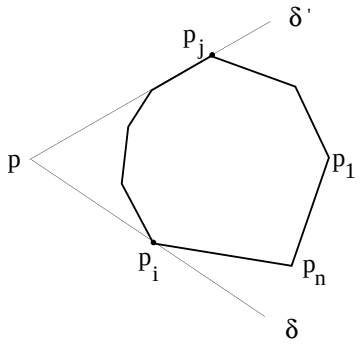

Figure 2.1: *Cône enveloppant.*

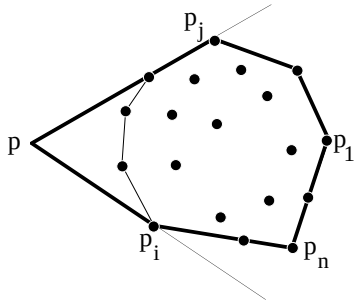

Figure 2.2: *Adjonction d'un point n'appartenant pas à l'enveloppe convexe.*

Problème 1. *Etant donné un ensemble fini S de points du plan, calculer $EC(S)$.*

Avant d'examiner quelques algorithmes classiques pour résoudre ce problème, nous pouvons donner une borne inférieure du temps de calcul.

Connaissant la nature de l'objet à calculer, précisons sous quelle forme nous voulons obtenir le résultat. Puisque $EC(S)$ est un polygone, nous demanderons que le résultat du calcul soit le contour positif du polygone, c'est à dire une suite circulaire de ses sommets.

Théorème 2.2. *Le calcul de l'enveloppe convexe de n points demande un temps $\Omega(n \log n)$.*

Preuve. Considérons un ensemble S de n points $p_i(x_i, x_i^2)_{i=1,\ldots,n}$ situés sur une parabole d'équation $y = x^2$. Le contour positif de $EC(S)$ permet donc d'obtenir, après une lecture, les nombres x_i triés par ordre croissant d'abscisses. Or le tri de n éléments d'un ensemble ordonné nécessite un temps $\Omega(n \log n)$. ∎

Notons que la preuve du théorème 2.1 donne un algorithme de calcul en temps $O(n^2)$.

11.2.2 Marche de Jarvis

La «marche de Jarvis» est un des algorithmes les plus naturels puisqu'il correspond à la technique d'emballage dite du «paquet cadeau» .

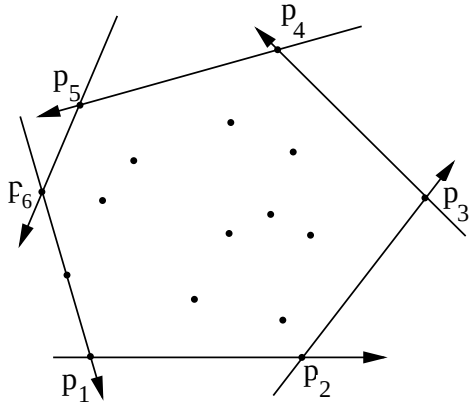

Figure 2.3: *Marche de Jarvis.*

Dans un premier temps, on peut chercher à déterminer non pas les sommets de l'enveloppe, mais ses arêtes. Or $[p, q]$ est une arête de l'enveloppe convexe si et seulement si tous les autres points sont situés sur le segment $[p, q]$ ou du même côté de la droite $d(p, q)$. On peut donc déterminer en temps $O(n)$ si un segment donné appartient à la frontière de l'enveloppe convexe. Or il y a $\binom{n}{2}$ segments à examiner. On obtient donc un algorithme en $O(n^3)$.

Jarvis a amélioré l'algorithme précédent en utilisant l'observation suivante : soit $((p_1, p_2, \ldots, p_n))$ le contour positif de l'enveloppe convexe des n points; le point p_{i+1} est le point minimal pour l'ordre polaire relatif à la demi-droite $d[p_i, \overrightarrow{p_{i-1}p_i}($, dans l'ensemble des sommets privé de p_i. Il suffit donc de déterminer deux points consécutifs p_1 et p_2 de l'enveloppe convexe, par exemple pour p_1 on peut choisir un point d'ordonnée minimale (et d'abscisse minimale, s'il y a plusieurs points d'ordonnée minimale) p_2 étant alors le point minimal dans $S - \{p_1\}$ pour l'ordre polaire relativement à la demi-droite $[p_1, \vec{\imath}($, ce qui prend un temps linéaire, puis de calculer les points p_i grâce à la remarque ci-dessus. D'où l'algorithme :

Algorithme JARVIS(S);
(1) Soit p_1 le point d'ordonnée minimale (et d'abscisse minimale s'il y a plusieurs points d'ordonnée minimale) de S;
(2) Soit p_2 le point minimal de $S - \{p_1\}$ pour l'ordre polaire par rapport à la demi-droite $[p_1, \overrightarrow{\imath}($;
(3) $k := 2$;
(4) répéter
(5) calculer le point q de S minimal pour l'ordre polaire par rapport à $[p_k, \overrightarrow{p_{k-1}p_k}($;
$k := k + 1$; $p_k := q$
jusqu'à ce que $p_k = p_1$.

Evaluons la complexité en temps de l'algorithme pour un ensemble S à n points. Clairement (1) et (2) prennent un temps $O(n)$. La boucle (4) est répétée $h - 1$ fois où h est le nombre de sommets de l'enveloppe convexe. La ligne (5) prend un temps $O(n)$. Donc l'algorithme est en temps $O(hn)$ où h est le nombre de sommets de l'enveloppe convexe, ce qui peut être intéressant si h est petit devant n, mais dans le pire des cas l'algorithme est en temps $O(n^2)$.

11.2.3 Algorithme de Graham

L'algorithme que nous exposons ici est dû à Graham; il repose sur les deux propriétés suivantes d'un polygone convexe :

Théorème 2.3. *Quelque soit le point O intérieur à un polygone convexe P, le contour positif de P est exactement le circuit polaire des sommets de P par rapport à O.*

Preuve. En effet si p, q, r sont trois points consécutifs du contour positif de P, alors on a $entre_O(p, q, r)$ du fait que P est convexe et que O est intérieur à P. ∎

On peut faire ici deux remarques :

— le résultat reste vrai si O est sur la frontière de P mais n'est pas un sommet de P;

— l'énoncé est faux pour un polygone quelconque, mais n'est pas caractéristique des polygones convexes (voir figure 2.4).

Théorème 2.4. *Un polygone simple est convexe si et seulement si pour tout triplet (p, q, r) de sommets consécutifs dans le sens direct, (p, q, r) est un tour gauche.*

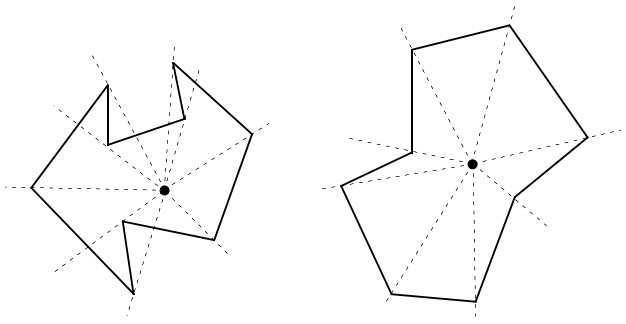

Figure 2.4: *Circuit polaire : exemple et contre-exemple pour un polygone non convexe.*

L'algorithme de Graham consiste à choisir un point intérieur à l'enveloppe convexe de S, mais n'appartenant pas à S, trier les points de S en un circuit polaire par rapport à O, et éliminer les points qui dans ce circuit ne constituent pas un tour gauche avec leurs deux voisins.

L'algorithme s'applique bien sûr dans le cas où les points de S ne sont pas alignés. Pour trouver un point intérieur à $EC(S)$ mais n'appartenant pas à S, il suffit de prendre l'isobarycentre O de trois points a, b, c de S non alignés, et de vérifier qu'il n'appartient pas à S. Si $O \in S$ alors on remplace O par le milieu de a et O, on itère ce processus au plus n fois. Donc trouver ce point O prend dans le pire des cas un temps $O(n)$.

On calcule ensuite le circuit polaire de S par rapport à O sous forme de liste doublement chaînée pour avoir accès en temps $O(1)$ au prédécesseur et au successeur d'un point du circuit. On réalise alors un parcours de cette liste $((p_0, \ldots, p_{n-1}))$ dans le sens croissant, en partant d'un point p_0 dont on est certain qu'il appartient à $EC(S)$, par exemple le point d'abscisse maximale (et d'ordonnée maximale en cas de non unicité) (fig.2.5). Si (p_{i-1}, p_i, p_{i+1}) est un tour droit, alors il est clair que p_i n'appartient pas à la frontière de $EC(S)$ puisque O est intérieur à $EC(S)$, donc p_i est intérieur au triangle $Op_{i-1}p_{i+1}$ et donc intérieur à $EC(S)$. L'algorithme consiste donc à parcourir la liste doublement chaînée en éliminant de tels points.

procédure GRAHAM(S) ;
(1) Déterminer un point O intérieur à S n'appartenant pas à S ;
(2) Calculer le circuit polaire de S par rapport à O ;
(3) Soit p_0 le point de S d'abscisse maximale (et d'ordonnée maximale s'il y en a plusieurs) ;
{ si p est un sommet, $succ(p)$ et $pred(p)$ désignent respectivement le successeur et le prédécesseur de p dans le circuit courant}
(4) $p := p_0$;
tantque $succ(p) \neq p_0$ faire
si $(p, succ(p), succ(succ(p))$ est un tour gauche alors $p := succ(p)$
sinon supprimer $succ(p)$; $p := pred(p)$
finsi
fintantque.

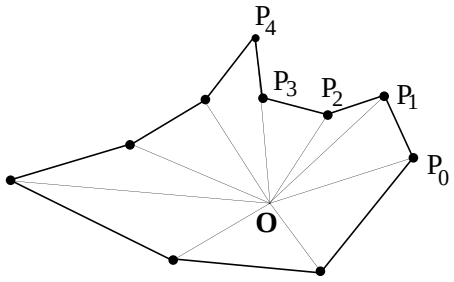

Figure 2.5: *Algorithme de Graham.*

Validité de l'algorithme

A la fin de la procédure on obtient une liste doublement chaînée de points de S telle que
(1) les points forment un circuit polaire par rapport à O ;
(2) trois points consécutifs quelconques forment un tour gauche.
On obtient donc la liste des sommets consécutifs d'un polygone convexe P en vertu des théorèmes 2.3 et 2.4, et puisque les points de S qui ont été supprimés n'appartenaient pas à la frontière de $EC(S)$, P est bien la frontière de $EC(S)$.

Théorème 2.5. *L'algorithme de Graham calcule l'enveloppe convexe de n points du plan en temps $O(n \log n)$ et en espace $O(n)$.*

Preuve. Les étapes 1 et 3 demandent, on l'a vu, un temps $O(n)$. L'étape 2 se réalise en temps $O(n \log n)$ (Proposition 1.5). Enfin il est aisé de voir que la boucle «tant que» (4) prend un temps linéaire. En effet, à chaque passage dans la boucle soit on avance d'un pas dans le parcours, soit on supprime un point. Or on ne peut avancer de plus de n pas, et on ne peut pas supprimer plus de n points.

Enfin l'espace nécessaire à l'exécution de la procédure est une liste doublement chaînée de taille n, c'est donc un espace linéaire. ∎

Traitons un exemple :

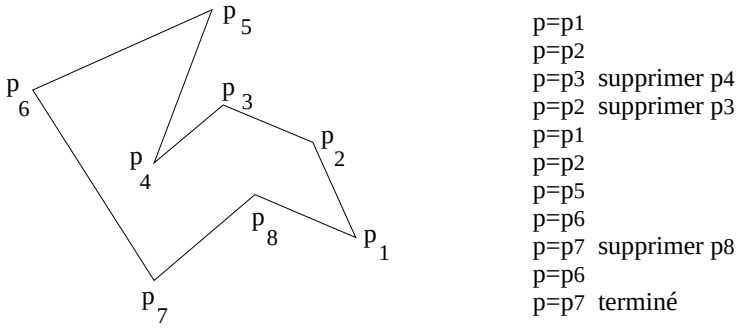

Figure 2.6: *Exemple.*

Bien que l'algorithme de Graham soit optimal dans le pire des cas, il est intéressant d'examiner d'autres algorithmes qui peuvent présenter des avantages variés.

11.2.4 Algorithme dichotomique

L'algorithme qui suit utilise le principe de dichotomie. Pour calculer l'enveloppe convexe $EC(S)$ d'un ensemble S, on divise S en deux parties S_1 et S_2 ayant le même nombre d'éléments à 1 près, on calcule leurs enveloppes convexes $EC(S_1)$, $EC(S_2)$, puis on détermine l'enveloppe convexe de $EC(S_1) \cup EC(S_2)$. L'efficacité de l'algorithme dépend de la complexité de la dernière étape. Or celle-ci consiste à calculer l'enveloppe convexe de l'union de deux polygones convexes, problème plus simple que celui d'origine que l'on peut espérer résoudre en temps linéaire. On est donc conduit à étudier le :

Problème 2. *Etant donnés deux polygones convexes P_1 et P_2, déterminer l'enveloppe convexe de leur union.*

Nous allons présenter un algorithme dont la complexité est donnée par le :

Théorème 2.6. *L'enveloppe convexe de l'union de deux polygones convexes se calcule en temps $O(N)$, où N est le nombre total de sommets.*

Preuve. On suppose qu'au moins un des deux polygones a plus de deux sommets (par exemple P_1, sinon on calcule en temps $O(1)$ l'enveloppe convexe des quatre points). Soit p un point intérieur à P_1, p est déterminé en temps $O(1)$.

Si p appartient à $EC(P_2)$ (ce qui se teste en temps $O(n_2)$, où n_2 est le nombre de sommets de P_2), alors les contours positifs de P_1 et P_2 sont respectivement les circuits polaires de P_1 et P_2 par rapport à p (théorème 2.3). En temps $O(n_1 + n_2)$

on peut fusionner ces deux circuits en un circuit polaire relativement à p. Il suffit ensuite d'appliquer l'étape 4 de l'algorithme de Graham qui est en temps $O(n_1 + n_2)$.

Si p est extérieur à P_2 (figure 2.7), on détermine en temps $O(n_2)$ deux sommets p_i et p_j de P_2 tels que (p_i, p, p_j) constitue le cône de sommet p qui enveloppe P_2 . On fusionne en temps $O(n_1 + n_2)$ le contour positif de P_1 et la liste des sommets de P_2 de p_j à p_i pour obtenir un circuit polaire de l'ensemble de ces points par rapport à p. Il n'y a plus qu'à appliquer l'étape 4 de l'algorithme de Graham à ce circuit, ce qui se réalise en temps $O(n_1 + n_2)$. ∎

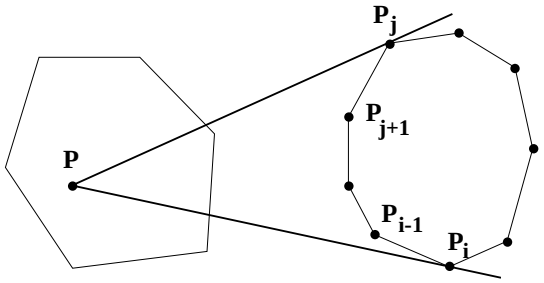

Figure 2.7: *Cas où p est extérieur à P_2.*

L'algorithme dichotomique s'écrit très simplement en procédant récursivement. Notons ENV_CONV_FUSION (P_1, P_2) la procédure qui calcule l'enveloppe convexe de l'union de deux polygones convexes P_1 et P_2. Alors :

```
fonction ENV_CONV_DICHO(S);
  si |S| ≤ 3 alors calculer directement l'enveloppe convexe de S
  sinon
    diviser S en deux ensembles S1 et S2 de même cardinal (à 1 près);
    P1 :=ENV_CONV_DICHO(S1);
    P2 :=ENV_CONV_DICHO(S2);
    ENV_CONV_FUSION(P1, P2)
  finsi.
```

Théorème 2.7. *En procédant par dichotomie, l'enveloppe convexe de N points se calcule en temps $O(N \log N)$.*

Preuve. Supposons que N est une puissance de 2. Soit $T(N)$ le temps de calcul de la fonction ENV_CONV_DICHO(S) pour un ensemble S à N éléments et $F(N)$ le temps de calcul de la procédure ENV_CONV_FUSION(P_1, P_2) où N est le nombre total de sommets de P_1 et P_2. On a alors

$$T(N) \leq 2T(N/2) + F(N) \tag{2.1}$$

Comme on l'a vu plus haut, $F(N)$ est $O(N)$. Donc de 2.1 on déduit que $T(N)$ est $O(N \log N)$. ∎

Notons que cet algorithme possède l'avantage de pouvoir être utilisé en parallèle en raison de son mécanisme de fusion.

11.2.5 Gestion dynamique d'une enveloppe convexe

Les algorithmes précédemment décrits opèrent tous de manière statique, c'est-à-dire qu'ils exigent de connaître l'ensemble S tout entier avant de commencer le calcul de l'enveloppe convexe de S. Si l'ensemble S est modifié par adjonction ou suppression de certains points, le calcul doit être repris complètement. La question qui se pose est donc la suivante : trouver un algorithme efficace pour calculer l'enveloppe convexe d'un ensemble dynamique de points. La réponse bien sûr n'est pas simple, comme on peut l'imaginer, en particulier la suppression d'un point de S peut faire apparaître sur la nouvelle frontière plusieurs points qui étaient auparavant internes à l'enveloppe convexe (figure 2.8). L'idéal serait

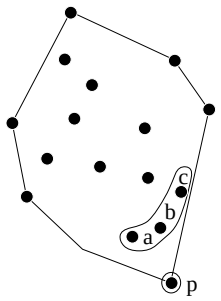

Figure 2.8: *Si p est supprimé, a, b et c apparaissent.*

d'obtenir un algorithme fournissant une représentation dynamique de l'enveloppe convexe d'un ensemble S de taille N tel que le temps de calcul de $EC(S)$ soit aussi bon que dans le cas statique, c'est-à-dire en temps $O(N \log N)$ et tel que la mise à jour de $EC(S)$ par adjonction ou suppression d'un point de S prenne un temps raisonnable, c'est-à-dire nettement inférieur à $N \log N$. Pour obtenir un temps de calcul de $EC(S)$ en $O(N \log N)$ il faudrait que la mise à jour par adjonction ou suppression d'un point prenne un temps $O(\log N)$. Aucun algorithme à ce jour ne donne ce résultat. Néanmoins l'algorithme que nous allons examiner offre un temps de mise à jour en $O(\log^2 N)$, ce qui donne un temps global de calcul de $EC(S)$ en $O(N \log^2 N)$ qui est bien sûr moins bon que dans un calcul statique, mais il répond à des exigences plus grandes.

La solution que nous proposons ici est due à Overmars et van Leeuwen. Une première chose à remarquer est la suivante : $EC(S)$ peut être considéré comme l'intersection de deux ensembles convexes non bornés (figure 2.9) qu'on appellera enveloppe convexe supérieure $ECS(S)$, et enveloppe convexe inférieure $ECI(S)$

que l'on peut décrire par abus comme les enveloppes convexes respectives des ensembles $S \cup \{P_{-\infty}(0, -\infty)\}$ et $S \cup \{P_{+\infty}(0, +\infty)\}$ respectivement.

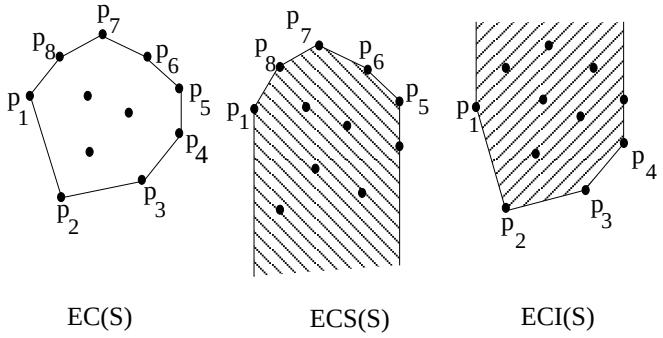

Figure 2.9: *Enveloppes convexes supérieure et inférieure.*

Les contours de ces deux enveloppes $ECI(S)$ et $ECS(S)$ donnent naissance à deux lignes polygonales qui convenablement concaténées définissent le contour direct de $EC(S)$. Dans l'exemple de la figure 2.9, on a :

$$\begin{aligned} ECS(S) &= (p_5, p_6, p_7, p_8, p_1) \\ ECI(S) &= (p_1, p_2, p_3, p_4) \\ EC(S) &= ((p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8)) \end{aligned}$$

Il nous suffit donc de nous limiter à l'étude de l'enveloppe convexe inférieure.

La structure de données choisie pour maintenir de manière dynamique l'enveloppe convexe $ECI(S)$ est, (ce n'est pas surprenant) une structure d'arbre, pour obtenir un coût de mise à jour réduit. On range les points de S aux feuilles d'un arbre binaire complet équilibré (c'est donc un arbre balisé au sens du chapitre 6). L'ordre pour la recherche est l'ordre lexicographique sur $\mathbb{R}^2$ (les points sont donc rangés par abscisses croissantes, puis en cas d'égalité par ordonnées croissantes). La balise pour piloter la recherche en un nœud v est un couple (x, y) représentant les coordonnées du point rangé dans la feuille la plus à droite du sous arbre gauche du nœud v. Chaque nœud v a pour vocation de représenter l'enveloppe convexe inférieure des points rangés dans les feuilles du sous-arbre de racine v, qu'on notera $Inf(v)$. Soient $filsG(v)$ et $filsD(v)$ les sommets de l'arbre qui sont respectivement fils gauche et fils droit du nœud v. Comment calculer $Inf(v)$ à partir de $I_1 = Inf(filsG(v))$ et $I_2 = Inf(filsD(v))$? La figure 2.10 donne le résultat.

Il existe un unique sommet p_1 de I_1 et un unique sommet p_2 de I_2 tels que le segment $[p_1, p_2]$ soit un côté de $Inf(v)$. On appellera *pont* ce segment, et p_1 et p_2 seront les deux *piliers* du pont. Le sommet p_1 scinde I_1 en deux lignes polygonales I_{11} et I_{12} (dans l'ordre), p_1 étant rangé dans I_{11}, et p_2 scinde I_2 dans l'ordre en deux lignes polygonales I_{21} et I_{22}, p_2 étant affecté à I_{22}. Alors $Inf(v)$ est la concaténation de I_{11} et I_{22}. Nous allons voir qu'il suffit de stocker en chaque sommet v la partie $Q(v)$ de l'enveloppe convexe inférieure $Inf(v)$ qui n'appartient pas à $Inf(père(v))$.

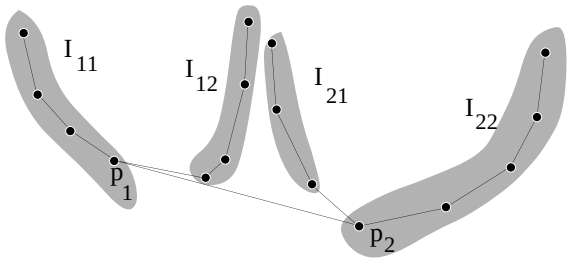

Figure 2.10: *Calcul de Inf(v) à partir de l'enveloppe convexe inférieure des feuilles des fils de v.*

Réciproquement, comment fabriquer *Inf*(*filsG*(v)) et *Inf*(*filsD*(v)) à partir de *Inf* (v) ? Il suffit pour cela de connaître les extrémités du pont $[p_1(v), p_2(v)]$ entre *Inf*(*filsG*(v)) et *Inf*(*filsD*(v)). A partir de *Inf*(v) et $p_1(v)$, on scinde *Inf*(v) en deux lignes polygonales qui, concaténées respectivement avec $Q($*filsG*$(v))$ et $Q($*filsD*$(v))$ deviennent les enveloppes *Inf*(*filsG*(v)) et *Inf*(*filsD*(v)) cherchées.

Ainsi nous pouvons décrire maintenant complètement la structure choisie. En chaque nœud v (différent d'une feuille) de l'arbre binaire sont stockées deux informations :

1) un pointeur vers la ligne polygonale $Q(v)$ stockée sous forme de liste concaténable (voir la section 6.5). Elle représente la partie de *Inf*(v) qui n'appartient pas à *Inf*($père(v)$) (si v est la racine, $Q(v) =$*Inf*(v)).

2) un couple $(p_1(v), p_2(v))$ représentant les piliers gauche et droit du pont de *Inf*(v).

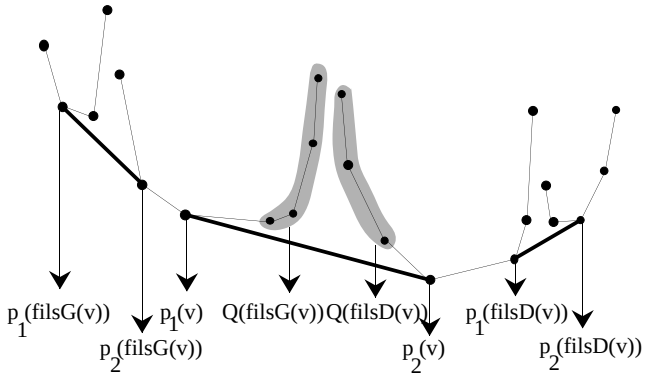

Figure 2.11: *Les fonctions p_1, p_2 et G.*

Cette représentation de l'enveloppe convexe inférieure d'un ensemble de points de taille N est intéressante car elle nécessite un espace $O(N)$. En effet, l'arbre a $2N-1$ sommets et les points rangés dans les listes $Q(v)$ en chaque nœud v sont au nombre total de N, puisque les listes $Q(v)$ forment en fait une partition de l'ensemble S.

Nous allons donner un algorithme qui calcule de manière efficace le pont $[p_1, p_2]$ entre I_1 et I_2 que l'on notera $\textsc{Pont}(I_1, I_2)$ et donc permet de calculer I à partir de I_1 et I_2.

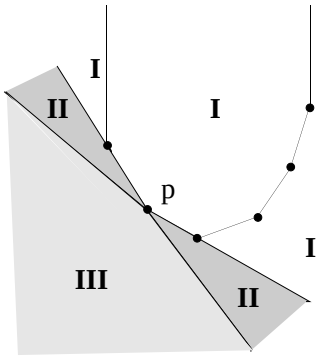

Figure 2.12: *Les 3 régions I, II, III auxquelles v peut appartenir.*

Il nous faut auparavant donner quelques définitions nécessaires à la mise en œuvre de l'algorithme. Soit p un sommet d'une enveloppe convexe inférieure $i = ECI(S)$ et v un point (fig. 2.12). Le point p est

- *concave* (relativement au segment $[p, v]$ et à i) si v est dans la région **(I)**;
- *tangent* si le point v est dans la région **(II)**;
- *réflexe* si le point v est dans la région **(III)**.

La région **(I)** est ouverte et la région **(III)** est fermée.

Soient I_1 et I_2 deux enveloppes convexes inférieures disjointes, i.e. séparées par une droite verticale d. Nous allons démontrer qu'avec la représentation précédemment définie, il est possible de calculer la fonction $\textsc{Pont}(I_1, I_2)$ efficacement .

Lemme 2.8. *Soient I_1 et I_2 deux enveloppes convexes inférieures disjointes. La fonction $\textsc{Pont}(I_1, I_2)$ se calcule en temps $O(\log N)$ où N est le nombre total de points.*

Preuve. Soient donc deux enveloppes convexes inférieures $I_1 = Inf(S_1)$ et $I_2 = Inf(S_2)$ disjointes (i.e. séparées par une droite verticale d). Nous pouvons alors déterminer le pont $[p_1, p_2]$ entre I_1 et I_2 qui permet de fabriquer l'enveloppe convexe $Inf(S_1 \cup S_2)$ comme étant l'unique segment $[p_1, p_2]$ reliant un sommet de I_1 à un sommet de I_2 et tel que p_1 soit un point tangent pour $[p_1, p_2]$ et I_2, et p_2 soit aussi un point tangent pour $[p_1, p_2]$ et I_1. Soit q_1 un sommet de I_1 et q_2 un sommet de I_2. Il y a 9 cas à envisager selon la nature respective des points q_1 et q_2 relativement au segment $[q_1, q_2]$ et relativement à I_2 et I_1 respectivement.

Ces 9 cas sont représentés sur la figure 2.13.

Sont hachurées les parties de I_1 et I_2 non candidates à contenir les piliers du pont. La figure est suffisamment explicite; seul le cas où q_1 et q_2 sont tous les

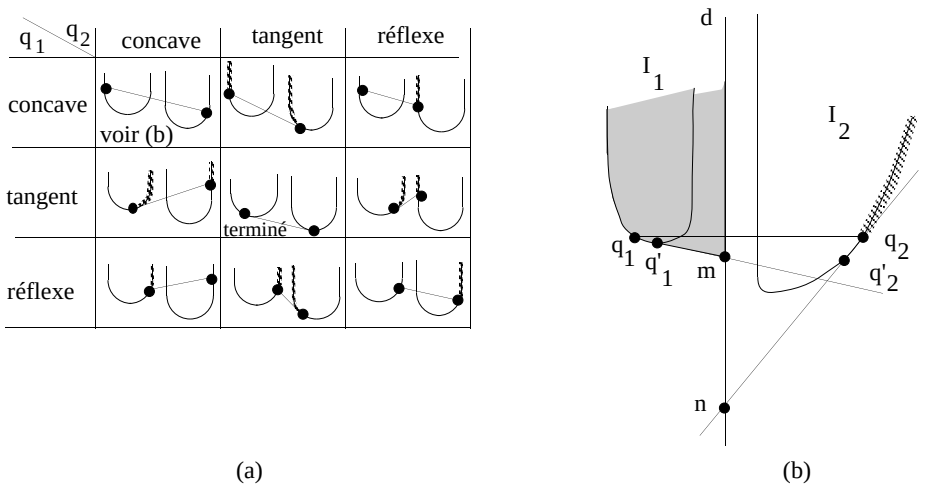

Figure 2.13: *Les* 9 *cas du lemme.*

deux concaves est un peu plus complexe. Il faut pour cela considérer la droite $d(q_1, q_1')$ où q_1' est le successeur de q_1 sur I_1 et la droite $d(q_2,\ q_2')$ où q_2' est le prédecesseur de q_2 sur I_2. Soit d une droite verticale séparant I_1 et I_2. La droite $d(q_1, q_1')$ (resp. $d(q_2, q_2')$) coupe d en m (resp. n). Si m est au-dessus au sens large de n (resp. au dessous au sens large) alors on peut hachurer la partie de I_2 à droite de q_2 (resp. la partie de I_1 à gauche de q_1). En effet pour tout point q'' de I_2 situé à droite de q_2 et tout point q' de I_1, le segment $[q', q'']$ est situé dans l'ensemble convexe limité par la partie de I_1 à gauche de q_1, les segments $[q_1, m]$, $[m, q_2]$ et la partie de I_2 à droite de q_2. Donc q'' est concave relativement à $[q', q'']$ et I_2, car $[q', q'']$ coupe d au-dessus de m alors que $[q_2, q_2']$ coupe d en n qui est au-dessous de m.

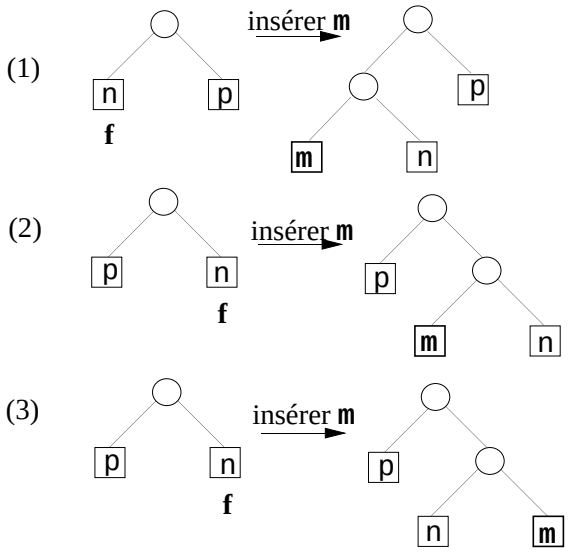

Figure 2.14: *Insertion d'un nouveau point* m.

Ainsi si le processus de recherche de $\textsc{Pont}(I_1, I_2)$ part des racines des arbres représentant les listes concaténables associées, ces arbres étant équilibrés, ils sont de hauteur au plus $\log N$ et $\textsc{Pont}(I_1, I_2)$ se calcule alors en temps $O(\log N)$. Notons qu'il faut prendre la précaution d'avoir accès dans la liste concaténable au voisin d'un sommet en temps $O(1)$. ∎

Il nous reste à décrire maintenant le processus de mise à jour de la structure de données décrivant de manière dynamique une enveloppe convexe inférieure, lors d'une insertion d'un nouveau point ou d'une suppression.

Prenons le cas d'une insertion. Les points sont rangés aux feuilles d'un arbre binaire complet équilibré. Il faut donc modifier les algorithmes classiques d'insertion : soit m le point à insérer et f la feuille à laquelle la descente aboutit en partant de la racine, f contenant le point n. Si f est une feuille gauche alors m est à gauche de n et on insère m selon le schéma (1) ; si f est une feuille droite, si m est à gauche de n alors on réalise (2) sinon on réalise (3) (figure 2.14).

```
procédure DESCENDRE(v, m);
  si v n'est pas une feuille alors
    (Q_G, Q_D) :=SCINDER(Inf(v), p_1(v), p_2(v));
    Inf(filsG(v)) :=CONCATÉNER(Q_G, Q(filsG(v)));
    Inf(filsD(v)) :=CONCATÉNER(Q(filsD(v)), Q_D);
    si m ≺ cle(v) alors DESCENDRE(filsG(v), m)
    sinon DESCENDRE(filsD(v), m)
    finsi;
  INSÉRER(m).
```

Préoccupons nous maintenant de la mise à jour des informations contenues dans les sommets de l'arbre en supposant pour l'instant, pour simplifier, que l'insertion réalisée ne demande pas de rééquilibrage.

La mise à jour se fait en deux temps. Lors de la descente dans l'arbre pour insérer une nouvelle feuille, pour chaque sommet v situé sur le chemin c de descente, on calcule $\mathit{Inf}(v)$ et $\mathit{Inf}(\mathit{frère}(v))$ inductivement et grâce aux points $p_1(v)$, $p_2(v)$ on transmet aux fils de v les lignes polygonales respectives qui leur manquent (figure 2.11) pour former l'enveloppe convexe qui leur est associée. Ainsi au moment de l'insertion du nouveau point, on dispose pour chaque sommet v du chemin c ainsi que pour ses frères, de l'enveloppe convexe correspondante.

Après insertion, la remontée à la racine (c'est ici que se situe un éventuel rééquilibrage) permet de recalculer inductivement $Q(v)$ pour tous les sommets v de c, et $p_1(v)$, $p_2(v)$ pour tous les sommets v de c ainsi que pour leurs frères (effectivement, on peut remarquer que la mise à jour des informations ne concerne que les sommets de c pour ce qui est de p_1 et p_2, mais concerne aussi les sommets frères pour ce qui est de Q). La mise à jour se réalise alors ainsi : si en un sommet v on

dispose de *Inf*(v) ainsi que de *Inf*$(frère(v))$ alors l'opération PONT permet de calculer $Q(v)$, $Q(frère(v))$ ainsi que *Inf*$(frère(v))$, $p_1(père(v))$ et $p_2(père(v))$, et donc inductivement permet la mise à jour des fonctions Q, p_1 et p_2. Ces deux opérations de descente et de montée sont décrites plus précisément dans les procédures DESCENDRE et MONTER, sans toutefois prendre en compte le rééquilibrage. Dans ces procédures, la variable v représente un sommet de l'arbre, et m est le point que l'on insère dans l'arbre.

```
procédure MONTER(v);
  si v ≠ racine alors
    (Q1, Q2, Q3, Q4, J1, J2) :=PONT(Inf(v), Inf(frère(v)));
    Q(v) := Q2;
    Q(frère(v)) := Q3; Inf(père(v)) :=CONCATÉNER(Q1, Q4);
    p1(père(v)) := J1, p2(père(v)) := J2;
    MONTER(père(v))
  sinon Q(v) :=Inf(v).
```

Examinons maintenant le mécanisme de rééquilibrage (par la méthode des arbres AVL). Prenons le cas d'une rotation droite par exemple (figure 2.15). Soient

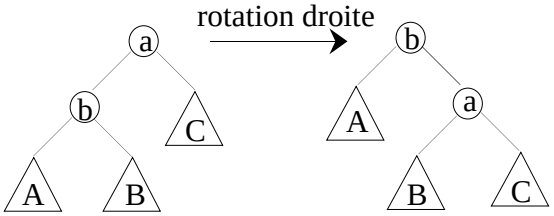

Figure 2.15: *Rotation droite.*

a', b', c' les racines des arbres A, B, C. La mise à jour des informations stockées dans les nœuds a, b, a', b', c' se fait par la suite d'instructions suivantes :

$(Q_1, Q_2, Q_3, Q_4, J_1, J_2)$ =PONT(*Inf*(B),*Inf*(C));
Inf(a) =CONCATÉNER(Q_1, Q_4); $Q(b') = Q_2$; $Q(c') = Q_3$;
$p_1(a) = J_1$; $p_2(a) = J_2$;
$(Q'_1, Q'_2, Q'_3, Q'_4, J'_1, J'_2)$ =PONT(*Inf*(A),*Inf*(a));
Inf(b) =CONCATÉNER(Q'_1, Q'_4); $Q(a') = Q'_2$; $Q(a) = Q'_3$;
$p_1(b) = J'_1$; $p_2(b) = J'_2$.

On peut remarquer que $Q(b)$ sera calculé au niveau de son père. Ainsi une rotation prend un temps $O(\log N)$.

Le procédé d'insertion est illustré par l'exemple ci-dessous (figures 2.16 et 2.17). La figure 2.16 représente l'arbre binaire équilibré dont les feuilles contiennent les points dont on doit calculer l'enveloppe convexe inférieure. On a représenté les points par des entiers qui correspondent à l'ordre dans lequel on a inséré les

points dans l'arbre. En chaque nœud v sont indiqués $Q(v)$ et $p_1(v)$. La figure 2.17 représente l'insertion d'un nouveau point. Les informations qui ont été recalculées au cours de cette insertion sont indiquées en gras et soulignées.

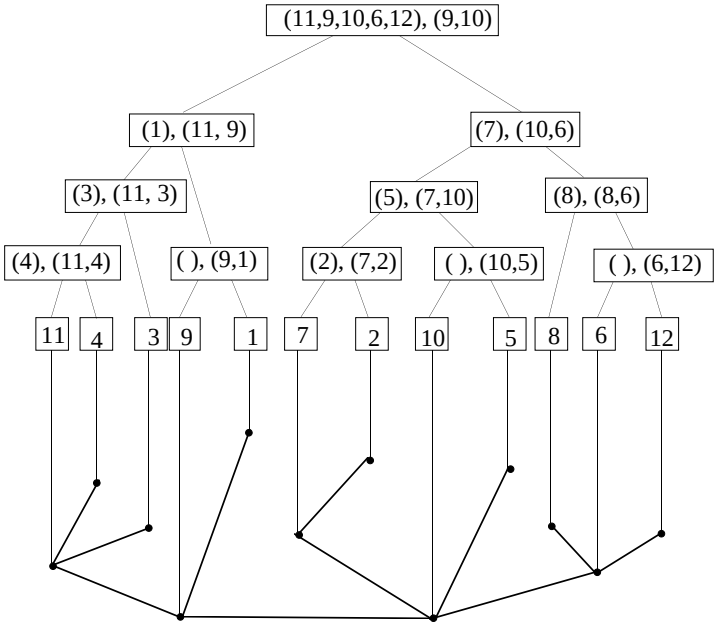

Figure 2.16: *Structure de l'arbre avant insertion du point* p_{13}.

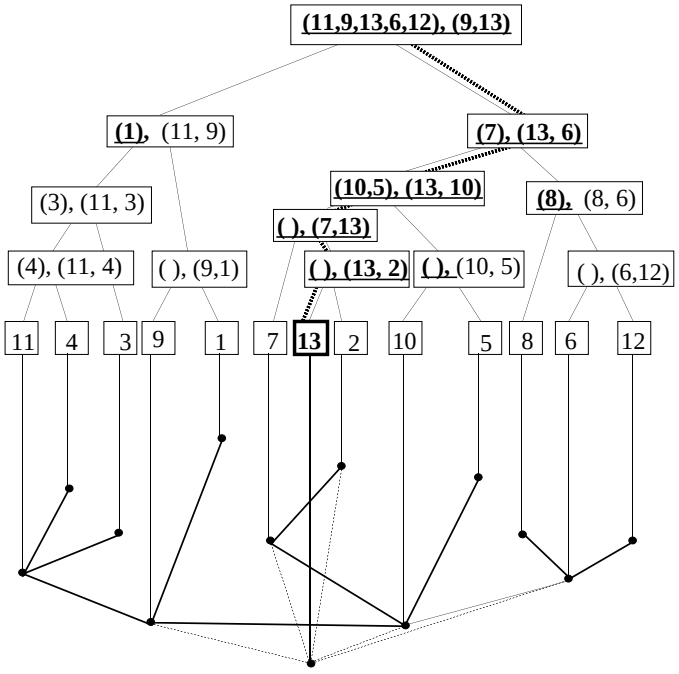

Figure 2.17: *Insertion du point* p_{13}.

Nous laissons au lecteur le soin de traiter le cas de la suppression d'un point

(exercice) qui est identique en ce qui concerne le mécanisme de mise à jour.

Analysons la complexité en temps de chacune de ces procédures. Rappelons que les opérations SCINDER et CONCATÉNER se réalisent en temps $O(\log k)$ où k est le nombre total d'éléments. Si N est le nombre de feuilles de l'arbre binaire, on a $k \leq N$; la hauteur de l'arbre étant $O(\log N)$, on en déduit que la procédure DESCENDRE, comme la procédure MONTER prennent un temps $O((\log N)^2)$. D'où le théorème :

Théorème 2.9. *Les enveloppes convexes inférieure et supérieure d'un ensemble de N points ont une mise à jour dynamique qui prend dans le pire des cas un temps $O((\log N)^2)$ par insertion ou suppression.*

Bien sûr cet algorithme donne un temps de calcul de l'enveloppe convexe de N points en temps $O(N(\log N)^2)$, ce qui est un temps moins bon que celui des algorithmes vus précédemment. Cela s'explique par le fait que la possibilité de mise à jour efficace par adjonction ou suppression d'un point nécessite l'utilisation de structures de données plus complexes, et donc un peu plus lourdes à manipuler.

11.3 Localisation de points dans le plan

Les problèmes de recherche en géométrie, bien que proches des problèmes de recherche classiques, présentent des particularités propres, liées précisément à leur nature géométrique et à la complexité des objets traités.

De manière très générale le problème de localisation est le suivant :

Données : L'espace est divisé en k régions $R_1, \ldots, R_k$.

Question : Soit m un point de l'espace. A quelle région appartient-il ?

La question est destinée à être posée pour un grand nombre de points. Il est donc intéressant de prétraiter les données de manière à obtenir un algorithme efficace pour la localisation de m. Les paramètres qui mesurent cette efficacité sont donc :

- le temps du prétraitement ;
- l'espace utilisé pour représenter les données ;
- le temps de réponse à la question.

Il y a encore peu de résultats sur ce problème en dimension 3 ou supérieure, par contre c'est un problème assez bien résolu dans le plan, du moins dans un cas simple, celui des subdivisions planaires. Il est certain que la nature de la partition détermine la complexité de la tâche.

Nous ne traiterons ici que le cas du plan. Considérons tout d'abord le problème élémentaire suivant :

Problème P_1 **:** Soit P un polygone simple (P divise le plan en deux régions). Etant donné un point z, z est-il intérieur à P ?

On peut encore simplifier le problème :

Problème P_2 **:** Soit P un polygone convexe. Un point z étant donné, le point z est-il intérieur à P ?

11.3.1 Cas d'un polygone simple

Un algorithme simple sans prétraitement permet de résoudre le problème P_1 en temps linéaire (en fonction du nombre de sommets du polygone).

Théorème 3.1. *Soit n le nombre de sommets d'un polygone simple P. On peut déterminer si un point est à l'intérieur de P sans prétraitement en temps $O(n)$.*

Preuve. On teste en temps $O(n)$ si z appartient au contour de P. Supposons donc que z n'appartient pas au contour de P. L'idée est simple : soit d une demi-droite d'origine z. Alors z est intérieur à P si et seulement si le nombre de points d'intersection de d avec le contour de P est impair ; en effet chaque fois que d coupe un côté de P, d passe de l'intérieur à l'extérieur de P ou viceversa et donc pour passer de z interne à P à « l'autre extrémité » de d qui est externe à P, il faut couper un nombre impair de côtés (figure 3.1). Tout le problème est de dénombrer

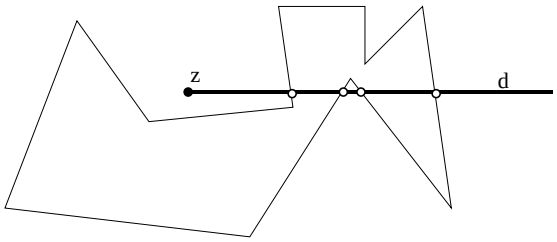

Figure 3.1: *Principe de l'algorithme.*

correctement les intersections en tenant compte des cas de dégénérescence. On voit clairement, dans les cas dégénérés indiqués dans la figure 3.2, quel est le nombre d'intersections qu'il faut compter. L'idée intuitive est qu'un déplacement « infinitésimal » de tous les points du polygone vers le haut ne modifie pas le résultat (i.e. le fait que z soit interne ou non à P). Ce déplacement infinitésimal permet d'éliminer les cas de dégénérescence. Les multiplicités indiquées sur le schéma sont réalisées lorsque l'on considère comme « intersectant d » uniquement les côtés de P non horizontaux qui rencontrent d mais dont l'extrémité inférieure a une ordonnée strictement inférieure à celle de z. D'où l'algorithme :

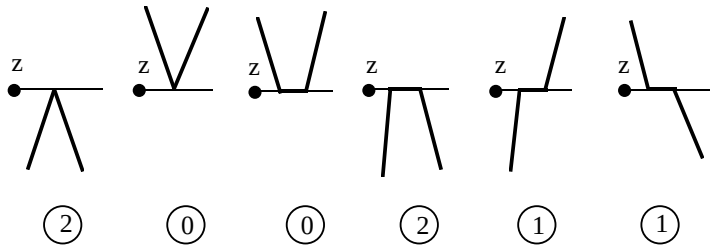

Figure 3.2: *Le décompte des intersections.*

procédure LOCALISER (z, P);
  Soit d une demi-droite horizontale d'origine z;
  $s := 0$;
  pour chaque côté non horizontal c de P faire
    si c rencontre d et l'extrémité inférieure de c est strictement
      inférieure à celle de z
    alors $s := s + 1$ finsi;
  si s est pair alors z est à l'extérieur de P sinon z est interne à P.

Il est clair que cet algorithme s'exécute en temps proportionnel au nombre de côtés de P.

11.3.2 Cas d'un polygone simple convexe

Nous allons voir maintenant comment traiter le problème P_2 en temps plus efficace grâce à un prétraitement qui utilise la convexité de P.

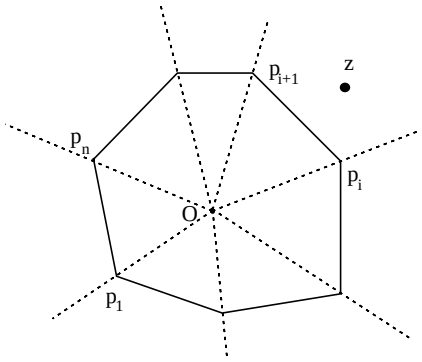

Figure 3.3: *Cas d'un polygone convexe.*

Théorème 3.2. *Soit P un polygone convexe ayant n sommets. Grâce à un prétraitement en temps $O(n \log n)$ on détermine si un point appartient à l'intérieur de P en temps $O(\log n)$ et en espace $O(n)$.*

Preuve. Soit $((p_1, \ldots, p_n))$ le contour direct de P. Si O est un point intérieur à P, le plan est divisé en n secteurs angulaires $(\overrightarrow{Op_i}, \overrightarrow{Op_{i+1}})$ (voir figure 3.3). Si on a déterminé à quel secteur angulaire z appartient, on détermine en temps $O(1)$ si z est intérieur à P ou non en repérant si z est à gauche ou à droite de la droite orientée $\overrightarrow{d}(p_i, p_{i+1})$. Le calcul du secteur angulaire se fait par dichotomie de la manière suivante :

```
procédure SECTEUR-ANGULAIRE(z, O, P);
  si z est entre p_n et p_1 alors terminé
  sinon i := 1; j := n;
    tantque j > i + 1 faire
      k := (i + j)div2;
      si z est entre p_i et p_k alors j := k
      sinon i := k
    fintantque
  finsi.
```

D'où l'algorithme complet :

```
procédure LOCALISER-POL-SIMPLE-CONVEXE(z, P);
(1) choisir un point intérieur à P;
(2) déterminer le secteur angulaire (Op_i, Op_{i+1}) contenant z;
(3) si z est à droite de p_i p_{i+1} alors
        z est externe à P
    sinon z est interne à P.
```

L'étape (1) prend un temps $O(1)$: il suffit de prendre le milieu de deux sommets non consécutifs de P.

L'étape (2) prend un temps $O(\log n)$ puisqu'on procède par dichotomie. La boucle «tant que» est exécutée $O(\log n)$ fois (on rappelle que la relation z est entre a et b équivaut à $(a \prec z$ et $z \prec b)$ ou $(z \prec b$ et $b \prec a)$ ou $(b \prec a$ et $a \prec z))$. L'étape (3) prend un temps $O(1)$. Enfin, l'implémentation de (2) peut se faire à l'aide d'un tableau et donc l'espace nécessaire est $O(n)$. ∎

11.3.3 Cas d'une subdivision planaire généralisée

Etudions maintenant le problème dans le cas d'une subdivision planaire quelconque.

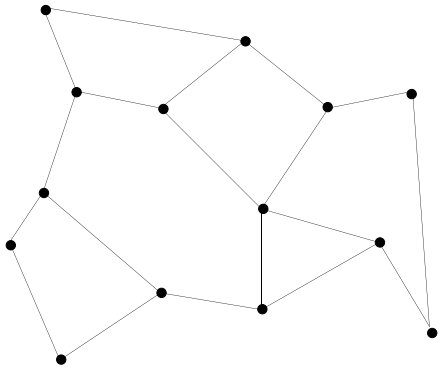

Figure 3.4: *Une subdivision planaire.*

Une *subdivision planaire* est une partition du plan en régions dont les contours sont des polygones simples, une et une seule de ces régions étant non bornée. Une *subdivision planaire généralisée* autorise les demi-droites et les droites comme côtés des polygones. La subdivision de la figure 3.5 possède 9 régions dont 3 sont bornées (deux côtés ne peuvent se couper qu'en une extrémité). Le problème de

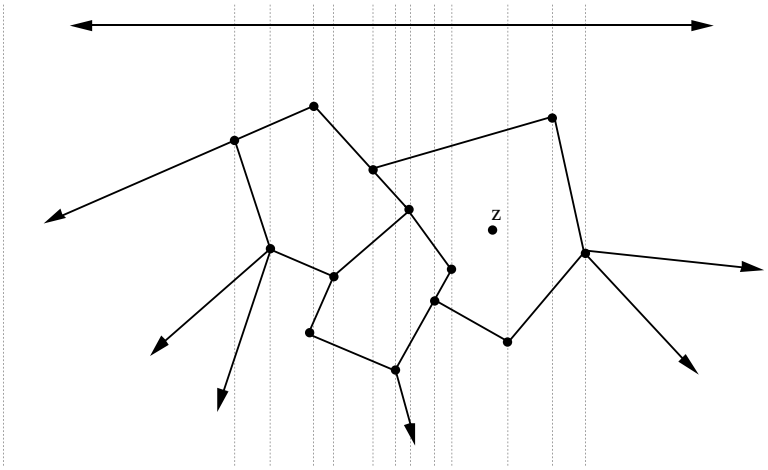

Figure 3.5: *Une subdivision planaire généralisée.*

localisation s'énonce donc de manière précise comme suit :

Données : Une subdivision planaire généralisée à n côtés.

Question : Si z est un point du plan, déterminer la région du plan à laquelle il appartient.

Le principe de l'algorithme que nous allons exposer est le suivant : par chaque sommet de la subdivision on fait passer une droite verticale; le plan est alors divisé en bandes verticales. Les côtés de la subdivision coupant une bande donnée peuvent être ordonnés de bas en haut (car deux côtés ne peuvent se couper à l'intérieur d'une bande). La localisation d'un point se ramène alors à deux recherches dichotomiques successives.

En temps $O(\log n)$ on détermine dans quelle bande est le point z, puis à l'intérieur de la bande on détermine en temps $O(\log n)$ entre quels segments le point z est situé, ce qui permet d'identifier la région.

Examinons alors soigneusement quel doit être le prétraitement. On suppose que la partition est donnée de telle sorte que pour chaque côté on connaisse en temps $O(1)$ les noms des deux régions situées de part et d'autre, et on suppose aussi que l'on connait le contour de chaque région.

Un prétraitement «brutal» consisterait à construire un arbre binaire de recherche pour chaque bande ce qui prendrait un espace $O(n^2)$ car chaque bande peut être coupée par $O(n)$ côtés. Mais on peut observer que les arbres binaires de recherche associés à deux bandes voisines comportent une certaine quantité d'information commune. Le prétraitement consiste donc à balayer le plan de gauche à droite par une droite verticale; la position initiale étant à $-\infty$, un arbre persistant contient les côtés de la subdivision coupés par la droite. La mise à jour se fait à chaque passage par le sommet suivant s de la subdivision (les sommets sont triés par ordre croissant d'abscisse). Au cours de cette mise à jour, on supprime les côtés d'extrémité droite s et on insère les côtés d'extrémité gauche s. Le prétraitement demande donc un temps $O(n \log n)$ et un espace $O(n)$.

Théorème 3.3. *Le problème de la localisation d'un point dans une subdivision à n côtés se résout en temps $O(\log n)$ par un prétraitement prenant un temps $O(n \log n)$ et un espace $O(n)$.*

On peut se demander si ce résultat est optimal. Si la partition se réduit à une bande horizontale formée de n rectangles, la localisation se ramène à la localisation d'un réel x dans une suite ordonnée $x_1 < \ldots < x_n$, ce qui prend un temps $\Omega(\log n)$ par une recherche binaire. Pour ce qui est du prétraitement, il nécessite clairement le rangement des données qui occupe un espace $O(n)$. Reste le temps du prétraitement qui peut être amélioré (pas dans cette version certainement puisqu'elle nécessite un tri préliminaire des sommets de la subdivision). Il existe un autre algorithme fondé sur la triangulation des polygones simples qui a un temps de prétraitement linéaire et qui conserve un espace linéaire et un temps de réponse logarithmique, mais il est beaucoup plus complexe et plus difficile à mettre en oeuvre.

11.4 Diagrammes de Voronoï

11.4.1 Diagrammes de Voronoï de points

Soit S un ensemble fini de points du plan P. Traditionnellement, les éléments de S sont appelés des *sites*.La *région de Voronoï* d'un site $a \in S$ est l'ensemble

$$R(a) = \{x \in P \mid d(a,x) \leq d(b,x) \text{ pour tout } b \in S\} \tag{4.1}$$

C'est donc l'ensemble des points du plan qui sont plus proches de a que de tout autre site. On peut donner une formulation plus compacte en introduisant les demi-plans

$$H_{a,b} = \{x \in P \mid d(a,x) \leq d(b,x)\}$$

pour $a, b \in S$, et $a \neq b$. Alors

$$R(a) = \bigcap_{b \neq a} H_{a,b}$$

Cette formule montre que la région de Voronoï $R(a)$ est convexe : c'est un *polytope convexe* (i. e. par définition une intersection finie de demi-plans). Une région de Voronoï est soit bornée (auquel cas c'est un polygone convexe), soit non bornée. Dans tous les cas, sa frontière est une ligne polygonale.

On appelle *diagramme de Voronoï* de S, et on note $\mathrm{Vor}(S)$, la réunion des frontières des régions de Voronoï de S. Le diagramme est formé de *sommets* qui sont les sommets des lignes polygonales des régions, et *d'arêtes*, qui sont les arêtes de ces lignes polygonales (voir figure 4.1). Comme cette terminologie le suggère,

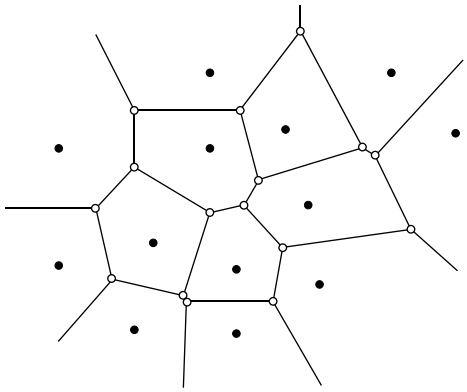

Figure 4.1: *Diagramme de Voronoï : les sites sont pleins.*

on peut aussi considérer le diagramme de Voronoï comme *graphe* abstrait, avec les sommets et les arêtes ainsi définis. Comme certaines frontières de régions de Voronoï ne sont pas bornées, les arêtes non bornées n'ont qu'une seule extrémité (mais on peut facilement remédier à cela en introduisant un sommet *à l'infini* dans la direction de l'arête).

Tout point x du diagramme de Voronoï est équidistant de deux sites au moins, et la distance de x à ces sites est minimale; donc $x \in \mathrm{Vor}(S)$ si et seulement s'il existe $a, b \in S$, $a \neq b$, tels que

$$d(a,x) = d(b,x) \leq d(c,x) \qquad \text{pour tout site} \quad c \in S$$

Un sommet du diagramme est un point qui est équidistant d'au moins trois sites à distance minimale. Revenons aux régions :

Proposition 4.1. *Une région de Voronoï $R(a)$ est non bornée si et seulement si a est un sommet de l'enveloppe convexe de S.*

Preuve. Supposons d'abord que $R(a)$ n'est pas bornée. Comme $R(a)$ est convexe, elle contient une demi-droite infinie D d'origine a. Si a n'est pas un sommet de l'enveloppe convexe de S, alors D intersecte l'enveloppe convexe en une arête $[b, c]$ de l'enveloppe convexe (voir figure 4.2). Mais alors tout point sur D qui est suffisamment loin, est plus proche de b (ou de c) que de a, contrairement à la définition de $R(a)$.

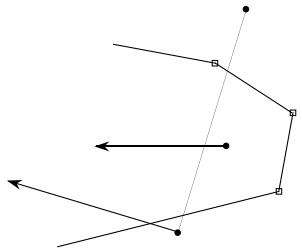

Figure 4.2: *Tout point suffisamment loin sur D est plus proche de b que de a.*

Réciproquement, si a est un sommet de l'enveloppe convexe de S, soient b et c ses voisins sur l'enveloppe. Tout point x qui est dans le secteur angulaire délimité par les demi-droites perpendiculaires aux segments $[b, a]$ et $[a, c]$ issues de a et orientées vers l'extérieur de l'enveloppe convexe (voir figure 4.3) appartient à la région de Voronoï $R(a)$, donc cette région n'est pas bornée. ∎

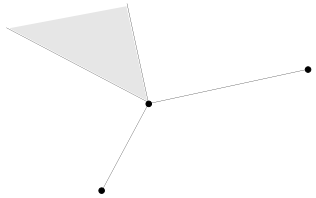

Figure 4.3: *La partie grisée est dans $R(a)$.*

Proposition 4.2. *Le diagramme de Voronoï* $\mathrm{Vor}(S)$ *d'un ensemble fini S est planaire en tant que graphe; de plus, si S ne contient pas quatre points cocycliques, tout sommet de* $\mathrm{Vor}(S)$ *est de degré* 3.

Preuve. Puisque $\mathrm{Vor}(S)$ est constitué des frontières des régions de Voronoï l'intersection de deux arêtes est par définition un sommet du graphe. Ceci prouve que le graphe est planaire.

Soit maintenant u un sommet de $\mathrm{Vor}(S)$, et soient $a_1, \ldots, a_k$ les sites tels que $d(u, a_1) = \cdots = d(u, a_k) \leq d(u, b)$ pour tout $b \in S - \{u, a_1, \ldots, a_k\}$. Les points $a_1, \ldots, a_k$ sont sur le cercle de centre u et de rayon $d(u, a_1)$, donc $k = 3$. ∎

Soit S un ensemble de sites, et soit x un point du plan. L'*amplitude*de x («clearance» en anglais) est le nombre

$$\delta(x) = \min\{d(x,a) \mid a \in S\}$$

C'est le rayon du plus grand disque de centre x et dont l'intérieur est disjoint de S. Ce disque est appelé le *disque de Delaunay* de x, et le cercle correspondant son *cercle de Delaunay*. On peut alors définir de manière équivalente le diagramme de Voronoï comme l'ensemble des points du plan dont le cercle de Delaunay contient au moins deux éléments de S, et les sommets du diagramme comme les points dont le cercle contient au moins trois sites.

Lorsque S ne contient pas quatre points cocycliques, on peut associer à S une triangulation appelée la triangulation de Delaunay, et que nous définissons maintenant. De manière générale, on appelle *triangulation* de S un ensemble de triangles vérifiant :

(1) la réunion des triangles est égale à l'enveloppe convexe de S;
(2) les intérieurs des triangles sont deux à deux disjoints;
(3) les sommets des triangles sont les éléments de S.

Soit p un sommet de $\mathrm{Vor}(S)$. Comme p est de degré 3, il existe exactement trois sites appartenant au cercle de Delaunay de p. Ces trois sites constituent les sommets d'un triangle $T(p)$ inscrit dans le cercle de Delaunay de p. Il est facile de vérifier que l'ensemble des triangles $T(p)$ constitue une triangulation de S appelée la *triangulation de Delaunay* de S.

11.4.2 L'algorithme de Fortune

Dans cette section, nous décrivons un algorithme pour calculer le diagramme de Voronoï d'un ensemble S fini de points (les «sites») dans le plan. L'algorithme est un algorithme de balayage. La construction est plus facile à comprendre lorsqu'on l'interprète dans l'espace $\mathbb{R}^3$. Le plan contenant S est le plan P d'équation $z = 0$. L'espace $\mathbb{R}^3$ est balayé par un plan d'équation $x + z = t$.

A chaque site $a \in S$ est associé un demi-cône C_a d'axe vertical (on parlera plus simplement de cône), défini par

$$C_a = \{(x,y,z) \mid d(a,(x,y)) = z\}$$

Ainsi, la hauteur z d'un point sur ce cône est égale à la distance de ce point à sa projection (x,y) sur le plan P.

Soient a et b deux sites, et C_a et C_b les cônes correspondants. Ces cônes s'intersectent en une branche d'hyperbole H. La hauteur z d'un point (x,y,z) sur cette hyperbole est égale à la fois à $d(a,(x,y))$ et à $d(b,(x,y))$; en d'autres termes, la projection (orthogonale) de H est la médiatrice du segment $[a,b]$. Autrement dit, cette hyperbole est l'intersection commune de C_a et de C_b avec le plan vertical défini par la médiatrice du segment $[a,b]$.

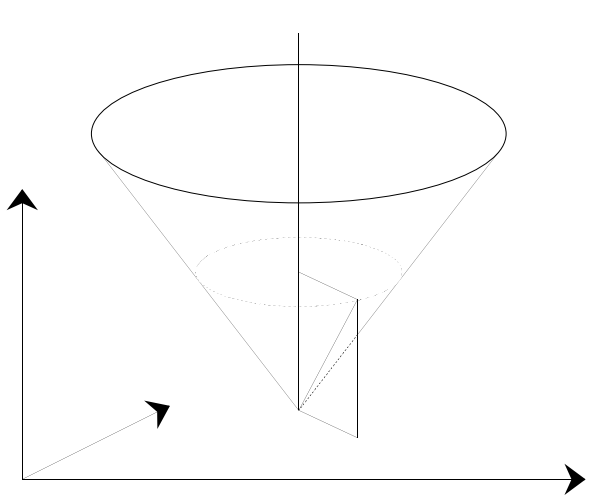

Figure 4.4: *Demi-cône associé au site a.*

Supposons maintenant que les cônes C_a pour $a \in S$, sont «opaques», et «regardons-les d'en bas». L'intersection de deux cônes associés à des sites a et b se projette en une droite, mais tous les points de cette droite ne sont pas visibles : un point p sera caché par un autre cône si le site de ce cône est plus proche de p que a et b ne le sont. Le segment de droite visible est donc délimité par des points projection de l'intersection de trois cônes.

Traduisons ceci en posant

$$C = \{(x, y, z) \mid z = \min_{a \in S} d(a, (x, y))\}$$

L'ensemble C est exactement la surface de tous les points visibles, puisque, pour chaque (x, y), on choisit le point de hauteur z minimale. Soit B l'ensemble des points de C qui appartiennent à au moins deux cônes. Alors on a :

Proposition 4.3. *La projection orthogonale de l'ensemble B des points qui appartiennent à au moins deux cônes de la famille C_a ($a \in S$) est égale au diagramme de Voronoï de S.*

Preuve. Soit $p = (x, y, z)$ un point de B. S'il appartient à la fois à C_a et à C_b pour deux sites distincts $a, b \in S$, alors le point $p' = (x, y)$ dans P est équidistant de a et de b, et cette distance est z. Donc (x, y) appartient au diagramme de Voronoï de S. Réciproquement, si $p' = (x, y)$ appartient au diagramme de Voronoï de S, on a $d(a, p') = d(b, p') = \min_{c \in S} d(c, p')$ pour deux points a, b de S. Le point $p = (x, y, d(a, p'))$ appartient à B. ∎

Front parabolique

L'algorithme de calcul du diagramme de Voronoï est un algorithme de balayage, par un plan oblique, qui est incliné de sorte à être tangent aux cônes quand il

passe par leur sommet. Plus précisément, à tout instant $t \in \mathbb{R}$, on considère le *plan de balayage* P_t d'équation $x + z = t$. Il intersecte donc le plan P contenant S en la droite Δ_t d'équation $x = t$. Appelons *ligne de balayage* cette droite.

Le plan balaye l'espace pour t croissant. L'intersection d'un cône C_a et du plan de balayage P_t est vide tant que le plan se trouve à gauche du cône, c'est-à-dire tant que t est plus petit que l'abscisse x_a de a. Au moment où le plan touche le cône, et est tangent au cône, l'intersection est une demi-droite passant par a; on peut voir cette demi-droite comme une parabole dégénérée. Ensuite, l'intersection est une parabole, et cette parabole s'«ouvre» au fur et à mesure que t croît. La projection de la parabole sur le plan P est une parabole $\pi_t(a)$ de foyer a, et de directrice Δ_t.

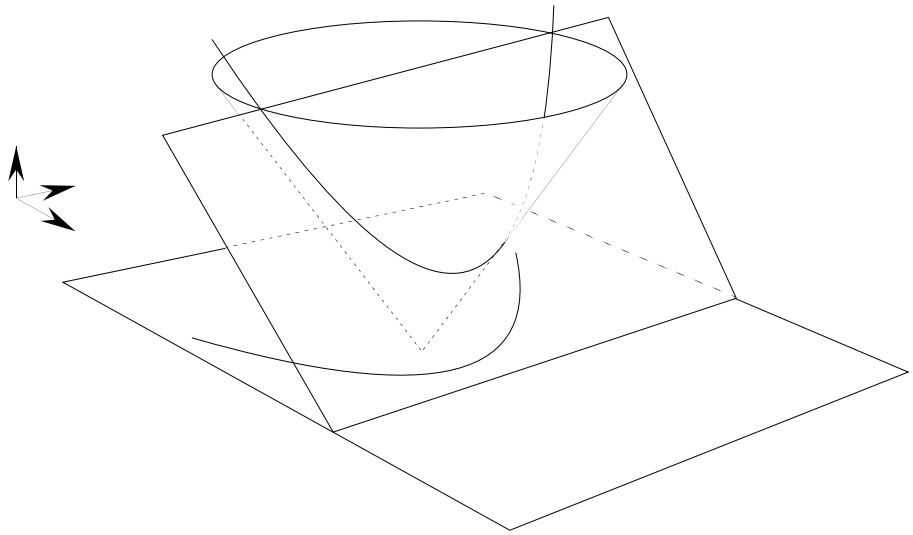

Figure 4.5: *Le plan de balayage.*

Supposons à nouveau les cônes C_a opaques, supposons de plus P_t opaque, et regardons à nouveau les surfaces d'en bas. Le plan P_t occulte tous les sites qui se trouvent à droite de Δ_t, donc les cônes associés à ces sites. Les cônes des sites déjà balayés donnent un ensemble de paraboles, toutes de même directrice, et de largeur variable. Seules certaines de ces paraboles sont en partie visibles, à savoir tant qu'elles ne sont pas entrées à l'intérieur d'autres cônes. Plus précisément, l'intersection de la surface C et du plan P_t se projette en une courbe F_t qui est une union d'arcs de parabole de même directrice Δ_t. Ces paraboles ont pour foyers des sites d'abscisse inférieure à t. La courbe F_t est appelée le *front parabolique* à

l'instant t.

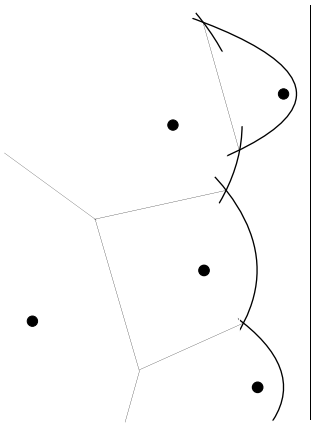

Figure 4.6: *Le front parabolique au temps t.*

Toute droite horizontale du plan P intersecte le front parabolique F_t en un point unique. Par ailleurs, F_t possède des *points anguleux*. Ce sont des points d'intersection de deux arcs de parabole consécutifs sur le front. Ils correspondent à la projection d'un point de B, et appartiennent donc au diagramme de Voronoï de S. La proposition suivante montre la réciproque.

Proposition 4.4. *Tout point du diagramme de Voronoï de S est point anguleux d'un front parabolique F_t pour un réel t.*

Preuve. Soit u une arête du diagramme de Voronoï de S, et soit p un point sur u. L'arête u est un intervalle sur la médiatrice de deux sites a et b. Posons $r = d(a,p) = d(b,p)$, et considérons la droite de balayage Δ_t située à droite de p et telle que $d(p,\Delta_t) = r$. En d'autres termes, Δ_t est tangente, en un point p', au cercle Π de centre p et de rayon r, cercle qui passe par a et b. Ce cercle ne contient, par définition du diagramme de Voronoï aucun autre site en son intérieur.

Le point p appartient aux paraboles $\pi_t(a)$ et $\pi_t(b)$. Supposons que p n'appartient pas au front parabolique F_t. Alors une autre parabole $\pi_t(c)$ passe strictement entre p et Δ_t, et coupe donc l'intervalle $]p,p'[$ en un point q. Or le site c est équidistant de p' et de q, donc se trouve sur le cercle Π' de centre q et de rayon qp'. Mais alors Π' est à l'intérieur de Π, et c est à l'intérieur de Π, ce qui est impossible. ∎

Il résulte de cette proposition qu'il «suffit» de balayer continûment l'espace par le plan P_t pour obtenir l'ensemble des points constituant le diagramme de Voronoï comme points anguleux des fronts paraboliques.

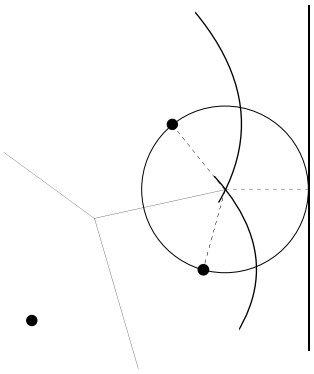

Figure 4.7: *Tout point du diagramme de Voronoï est point anguleux.*

Evénements

En fait, comme le diagramme de Voronoï est formé d'un nombre fini de points et de segments, on peut se contenter de calculer un nombre fini de données. De manière analogue, la suite des sites relatifs aux arcs du front parabolique ne change qu'en un nombre fini d'instants t, lorsque t varie de $-\infty$ à $+\infty$; ce sont ces instants qui constituent les «événements» dont nous parlons maintenant. Un *événement* est un instant t où un arc de parabole apparaît ou disparaît sur le front parabolique F_t.

Apparition d'un nouvel arc de parabole

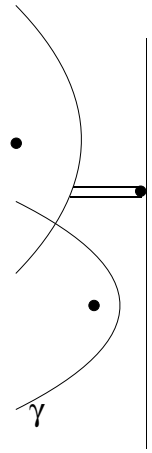

Figure 4.8: *Apparition d'un nouvel arc α.*

Lorsque la droite de balayage Δ_t passe par un site a, une nouvelle parabole $\pi_t(a)$ apparaît. Au départ, cette parabole est dégénérée, et est réduite à une demi-droite horizontale issue de a. Cette parabole (demi-droite) intersecte le front parabolique en un seul arc, ou alors en deux arcs consécutifs ; dans ce cas, le

point d'intersection est un point anguleux. Ces situations sont les seules où peut apparaître un nouvel arc parabolique :

Proposition 4.5. *Un nouvel arc parabolique apparaît à l'instant t dans le front parabolique si et seulement si Δ_t balaye un nouveau site.*

Preuve. Supposons le contraire. Alors un nouvel arc parabolique α apparaît dans le front parabolique à un instant t sans être créé par l'arrivée d'un nouveau site. La parabole π dont α est un arc existait donc déjà, mais était auparavant située entièrement à gauche du front parabolique. Au moment de l'apparition de l'arc α dans le front, la parabole π est

- soit tangente à un arc de parabole du front ;
- soit passe par un point anguleux du front.

Les deux situations mènent à une contradiction. En effet, si π est tangente à un arc β d'une parabole π', cela signifie que π a un rayon de courbure inférieur à celui de π', donc que son foyer est plus proche de Δ_t que celui de π'. Or π est contenue (au sens large) dans π', donc son sommet est plus éloigné de Δ_t que le sommet de π'. Ces deux conditions sont contradictoires.

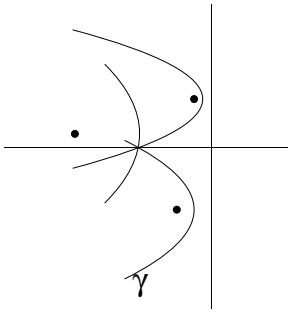

Figure 4.9: *Apparition de l'arc α : deuxième cas impossible.*

Pour montrer que le deuxième cas est également impossible, il nous faut un peu de calcul, et des notations (voir figure 4.9). Notons (x_s, y_s) les coordonnées d'un point s. Soient β et γ les deux arcs de parabole dont l'intersection définit le point anguleux $p = (x_p, y_p)$, à l'instant t où l'arc de parabole α apparaît. Soient b et c les foyers des paraboles supportant β et γ, et soit a le foyer de π. On suppose $y_b > y_p > y_c$. On a alors $x_a \leq x_b$ ou $x_a \leq x_c$, selon que $y_a \geq y_p$ ou $y_a \leq y_p$.

Considérons la droite horizontale H d'équation $y = y_p$. Au temps t, les paraboles $\pi_t(a)$, $\pi_t(b)$, et $\pi_t(c)$ intersectent H au point p. Au temps $t + \varepsilon$, pour $\varepsilon > 0$ assez petit, l'intersection de H avec $\pi_{t+\varepsilon}(a)$ se situe à droite de l'intersection de H avec $\pi_{t+\varepsilon}(b)$, et à droite de $\pi_{t+\varepsilon}(c)$. Ceci signifie qu'au temps t, la vitesse avec laquelle le point d'intersection $\pi_t(a) \cap H$ se déplace sur H est supérieure à la vitesse correspondante de $\pi_t(b) \cap H$. Calculons donc cette vitesse. L'équation d'une parabole $\pi_t(s)$ de foyer s et de directrice Δ_t est

$$(x - x_s)^2 + (y - y_s)^2 = (x - t)^2$$

ou encore $2x(t-x_s) = t^2 - x_s^2 - (y-y_s)^2$. Soit $x^{(a)}(t)$ l'abscisse du point $\pi_t(a) \cap H$. En dérivant, on obtient $2x^{(a)} + 2(dx^{(a)}/dt)(t - x_a) = 2t$, soit encore

$$\frac{dx^{(a)}}{dt} = \frac{t - x^{(a)}}{t - x_a}$$

et des formules analogues pour b et c. Au temps t, on a $x^{(a)}(t) = x^{(b)}(t) = x^{(c)}(t) = x_p$. Comme

$$\frac{dx^{(a)}}{dt} > \frac{dx^{(b)}}{dt}$$

on a donc $x_a < x_b$, et de même $x_a < x_c$, une contradiction. ∎

Disparition d'un arc de parabole

Au moment où un arc de parabole β disparaît, il est réduit à un point p qui est en fait l'intersection de trois arcs de parabole consécutifs α, β et γ. Par un argument déjà employé plus haut, on vérifie que α et γ n'appartiennent pas à la même parabole. Le point p est équidistant des foyers a, b et c des paraboles correspondantes, et de la droite de balayage Δ_t. En d'autres termes, le point p est un sommet du diagramme de Voronoï. Plus précisément, Δ_t est tangent au cercle circonscrit aux sites a, b, c de centre p, qui est un cercle de Delaunay. Nous venons de prouver :

Proposition 4.6. *Un arc disparaît du front parabolique à l'instant t si et seulement si Δ_t est tangent à un cercle de Delaunay centré sur un sommet du diagramme de Voronoï.*

Deux structures de données sont nécessaires pour mettre en œuvre l'algorithme de Fortune. Ces structures de données sont d'ailleurs toujours de même nature lorsque l'on veut réaliser un algorithme de balayage : une première structure gère les événements. Chaque événement correspond à un nombre réel, qui est l'abscisse de la droite de balayage. On extrait les éléments en ordre croissant, et on insère si nécessaire de nouveaux éléments ; il convient donc d'employer une *file de priorité*. La deuxième structure gère les objets actifs, ici les arcs du front parabolique. Ces objets sont naturellement ordonnés par ordonnées croissantes, et les opérations à effectuer sont celles d'un *dictionnaire*. On réalise donc l'implémentation en faisant appel à un arbre de recherche équilibré. Si le nombre d'événements est linéaire en fonction du nombre n de sites, il y a de fortes présomptions pour que l'algorithme coûte au total $O(n \log n)$ opérations, et une place $O(n)$. On verra qu'il en est ainsi aussi pour l'algorithme de Fortune.

File des événements

La file des événements, notée $\mathcal{E}$, contient deux types d'événements : les événements de type *site*, et les événements de type *cercle*. Chaque événement est décrit

par son type, et par un nombre réel, qui est l'instant t où il doit être extrait de la file et traité. Un événement de type site est l'arrivée de la ligne de balayage sur un nouveau site. La figure 4.10 montre un tel événement. Ces événements sont repérés et rangés par les abscisses croissantes des sites. Un événement de type cercle correspond à la disparition d'un arc de parabole. Comme nous l'avons vu plus haut, la disparition se produit au moment où trois arcs de parabole s'intersectent en un point anguleux. Ce point anguleux est alors le centre d'un cercle de Delaunay tangent à la droite de balayage à cet instant, et c'est un sommet du diagramme de Voronoï. C'est ainsi que sont créés les sommets du diagramme de Voronoï. Un événement de type cercle est repéré par la plus grande valeur t de la droite de balayage Δ_t tangente au cercle, puisque c'est à cet instant qu'un arc de parabole disparaît du front parabolique.

Au début de l'exploration, la file $\mathcal{E}$ est initialisée avec les n événements sites. Au fur et à mesure du balayage, des événements du type cercle sont créés et insérés dans la file. Plus précisément, des événements de type cercle sont créés (voir aussi plus bas) chaque fois que trois arcs de parabole deviennent consécutifs. Lors de la création de tels événements, on n'est pas certain qu'ils donneront naissance à un sommet du diagramme de Voronoï, car les sites à droite de la ligne de balayage ne sont pas encore connus à cet instant. On les insère quand même dans la file, quitte à les supprimer ultérieurement.

Voici donc une description de l'algorithme de Fortune, qui sera détaillée dans la suite :

```
procédure FORTUNE;  (description sommaire)
  Insérer les événements sites dans la file E;
  tantque la file E n'est pas vide, faire
    extraire un événement p;
    si l'événement est un site, alors
      créer un nouvel arc parabolique, et une arête de Voronoï;
      désactiver les événements cercle obsolètes;
      insérer les événements cercle;
    si l'événement est un cercle, alors
      supprimer l'arc parabolique, créer un sommet de Voronoï;
      désactiver les événements cercle obsolètes;
      insérer les événements cercle;
  fintantque.
```

Arbre du front parabolique

Les arcs de parabole constituant le front parabolique sont naturellement ordonnés par valeur croissante de leurs ordonnées. On les range aux feuilles d'un arbre

binaire de recherche équilibré $\mathcal{F}$, qui contient quelques informations complémentaires. Un nœud x de l'arbre contient un couple (a, b) de sites, à savoir les sites des arcs de parabole extrêmes de l'intervalle de paraboles aux feuilles du sous-arbre de racine x. Ces informations permettent de piloter la recherche de l'arc de parabole intersectant une droite horizontale d'ordonnée donnée. En fait, l'arbre $\mathcal{F}$ est enrichi comme suit :

- chaque feuille (représentant un arc du front) contient un pointeur vers le site associé ;
- les feuilles sont chaînées entre elles, mais indirectement : entre deux feuilles consécutives est placé un pointeur (que nous appelons *chaînon*) vers une arête du diagramme de Voronoï en cours de construction.

En effet, comme nous l'avons montré plus haut, le point anguleux entre deux arcs de parabole consécutifs du front est sur une arête du diagramme de Voronoï et c'est un pointeur vers cette arête, dont on ne connaît d'ailleurs en général pas les extrémités, qui est le chaînon entre deux feuilles successives. Cela signifie aussi que de chaque feuille, on peut accéder en temps constant aux voisins. Nous avons vu que ceci est une extension facile des arbres de recherche balisés. Une arête elle-même contient un couple de pointeurs vers (le ou) les sommets du diagramme de Voronoï qui sont ses extrémités. Comme on le verra, la création d'une arête et l'attribution des extrémités à l'arête ne se font pas nécessairement au même instant.

Gestion du front

Les événements sont extraits dans l'ordre de la file des événements $\mathcal{E}$. Voici les opérations à réaliser sur l'arbre.

Si l'événement est un site a, un arc parabolique dégénéré α est créé. On descend dans l'arbre $\mathcal{F}$ à la recherche de l'arc parabolique, disons β, qui est intersecté par α (le cas où α intersecte le front en un point anguleux sera considéré plus loin). L'arc β est coupé en deux arcs β' et β'' entre lesquels vient se placer α. Dans l'arbre $\mathcal{F}$, on substitue donc, à la feuille β, trois feuilles β', α, β'', et on insère les deux nœuds nécessaires. Une nouvelle arête v du diagramme de Voronoï est créée, et les deux chaînons entre β' et α d'une part, entre α et β'' d'autre part, pointent vers cette arête (on ne connaît pas encore les sommets du diagramme de Voronoï qui seront les extrémités de l'arête).

Si l'événement est du type cercle, cela signifie d'abord la suppression d'un arc de parabole β réduit à un point s qui est un sommet du diagramme de Voronoï. La suppression de la feuille β dans l'arbre $\mathcal{F}$ (avec le rééquilibrage éventuellement nécessaire) est donc accompagnée de la création d'un sommet du diagramme de Voronoï. De plus, les arêtes sur lesquelles pointent les deux chaînons voisins de β reçoivent leur extrémité correspondante : c'est le point s. La suppression de β et de ses deux chaînons est suivie de l'insertion d'un chaînon entre ses feuilles voisines. Ce chaînon pointe vers une nouvelle arête, disons w, dont une extrémité est le sommet s.

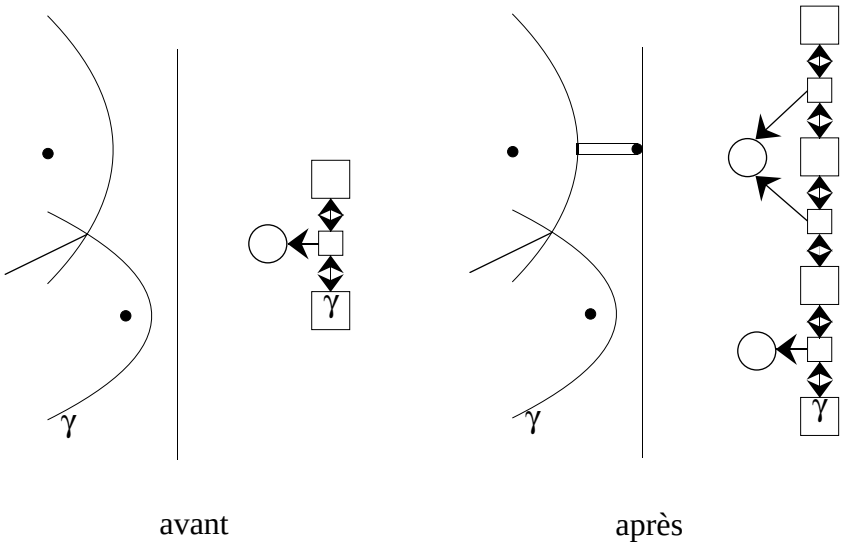

Figure 4.10: *Un événement de type site : cas «général».*

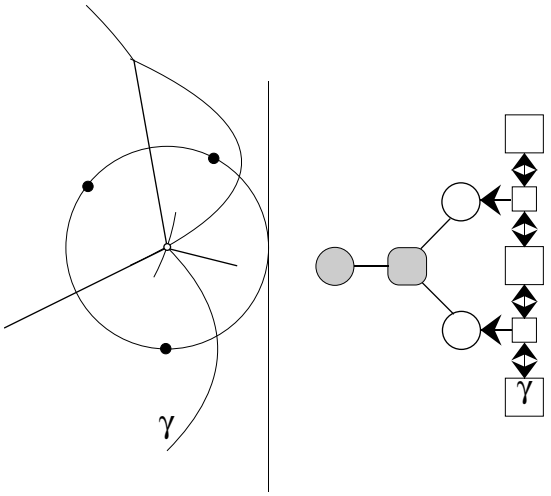

Figure 4.11: *Disparition de β et création du sommet s.*

Toutes ces opérations (sauf la recherche et le rééquilibrage) se font bien sûr en temps constant.

Reste le cas d'un événement de type site «spécial» : c'est le cas où l'arc parabolique dégénéré α créé par l'arrivée du site a intersecte le front parabolique en un point anguleux (voir figure 4.12). Dans ce cas, les deux opérations ci-dessus se succèdent : l'arc α est inséré dans l'arbre $\mathcal{F}$ entre les deux arcs qu'il intersecte, et le sommet s du diagramme de Voronoï est créé immédiatement.

Gestion des événements

Comme nous l'avons dit, les événements de type site sont créés au début des

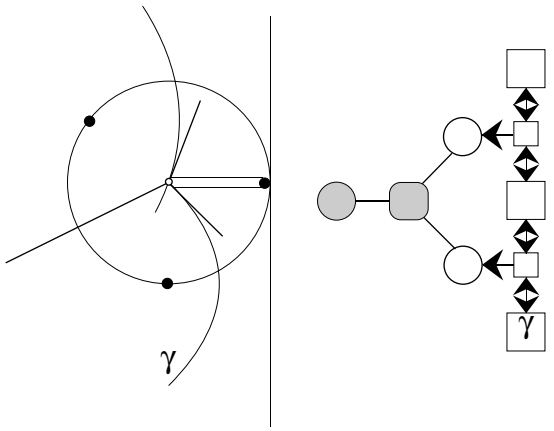

Figure 4.12: *Evénement site avec création d'un sommet.*

calculs. Les événements de type cercle signalent un sommet du diagramme de Voronoï, ou de manière équivalente, un cercle de Delaunay contenant 3 sites au moins. Un tel événement est créé chaque fois que l'on rencontre, pour la première fois, trois arcs consécutifs du front parabolique. Dans ce cas, le cercle circonscrit à leurs sites est calculé. Cet événement est repéré par l'abscisse maximale d'un point sur ce cercle. La gestion de ces événements est plus délicate : chaque fois qu'un nouveau site est introduit, il peut apparaître à l'intérieur de cercles calculés précédemment, et dont on avait supputé, au moment de leur création, qu'ils étaient des cercles de Delaunay. Pour tenir compte de cette situation, il faudra détruire, ou désactiver, des événement de type cercle introduits dans la file (on suppose qu'un événement de type cercle possède un champ qui signale s'il est actif ou non). Un événement de type cercle est déterminé par trois arcs consécutifs α, β, γ du front parabolique. On dira qu'il est *associé* à β; l'événement est actif tant qu'il continue à être associé à un arc parabolique.

La création d'événements de type cercle se fait lors du traitement des événements de $\mathcal{E}$. Considérons les deux cas.

Un événement de type site provient de la rencontre d'un site a; considérons d'abord le cas «général» : on crée un arc parabolique (dégénéré) α qui coupe en deux arcs β', β'' un arc β (voir ci-dessus). L'événement de type cercle associé à β, s'il existe, est détruit (désactivé), et deux nouveaux événements de type cercle sont crées, s'il y a assez d'arcs, l'un associé à β', et l'autre associé à β''. Notons que comme a est sur les cercles correspondants, ces événements se situent à droite de la ligne de balayage. Dans le cas d'un événement de type site «spécial», on supprime les événements de type cercle associés aux voisins β et γ de α, et on crée de nouveaux événements de type cercle pour β et γ (avec α comme voisin!).

Si l'événement est de type cercle, il y a disparition d'un arc β du front parabolique (précisément celui auquel est associé l'événement en cours de traitement!). Soient

α et γ les arcs qui précèdent et suivent β (à supposer qu'ils existent). Les éventuels événements de type cercle associés à ces arcs sont désactivés, et d'autres sont calculés : soit δ l'arc qui suit γ dans le front (s'il existe). Les trois arcs α, γ, δ sont maintenant consécutifs; soit C le cercle circonscrit à leurs sites. Si l'abscisse maximale d'un point de C est supérieure à la valeur t de balayage, l'événement ainsi créé est associé à γ et inséré dans la file; sinon, il est ignoré.

Expliquons pourquoi on peut ignorer l'événement si l'abscisse maximale d'un point de C est inférieure à t : dans ce cas, si C *est* un cercle de Delaunay, son centre est un sommet du diagramme de Voronoï, et il a donc déjà été traité antérieurement (en d'autres termes, un événement de type cercle déterminé par α, γ, et δ a déjà été créé et traité antérieurement).

Analyse

Il nous reste à analyser le temps pris par l'algorithme :

Théorème 4.7. *L'algorithme de Fortune calcule le diagramme de Voronoï de n sites en temps $O(n \log n)$ et en place $O(n)$.*

Preuve. Chaque événement de type site crée un arc parabolique dégénéré, et sauf dans un cas spécial, coupe en deux un arc parabolique. Un tel événement crée donc trois arcs, et en supprime un (celui qui est coupé en deux). Un événement de type cercle ne crée pas d'arc. Le nombre *total* d'arcs de parabole présents durant le balayage est $O(n)$. Le traitement d'un événement cercle crée un sommet du diagramme de Voronoï, et comme le diagramme est planaire, le nombre d'événements cercles traités est en $O(n)$. Mais tous les événements créés ne sont pas traités, puisque certains sont supprimés auparavant. Toutefois, le traitement d'un événement site ou cercle ne crée qu'un nombre constant d'événements, donc le nombre d'événements de type cercle est aussi en $O(n)$.

Ainsi, le nombre total d'événements créés est en $O(n)$. La place nécessaire pour gérer ces événements, et pour gérer l'arbre des arcs de parabole, est donc $O(n)$. Chaque événement requiert $O(\log n)$ opérations, pour l'insertion dans la file, et chaque gestion d'arc de parabole demande également un temps $O(\log n)$. D'où le temps total annoncé. ∎

11.4.3 Diagramme de Voronoï de segments

Dans cette section, on considère des diagrammes de Voronoï de segments. Il existe plusieurs définitions; nous choisissons de présenter celle qui est la mieux adaptée au problème de planification de trajectoires dont il sera question dans le chapitre suivant. Etant donné un segment $s = [a, b]$ du plan, on appellera, par abus de langage, *segment ouvert* le segment s privé de ses extrémités a et b.

Soit S un ensemble formé d'un nombre fini de *segments ouverts* deux à deux disjoints, et d'un ensemble fini de *points*. Les points qui sont les extrémités des

segments sont toujours supposés appartenir à S. Pour tout segment ouvert s, on notera $\bar{s}$ le segment formé de s et de ses extrémités. On appellera indistinctement *primitive* un point ou un segment ouvert de S. L'ensemble de la figure 4.13 ci-dessous est formé de 10 primitives, les 6 points a, b c, d, e, f et les 4 segment ouverts u, v, w, t.

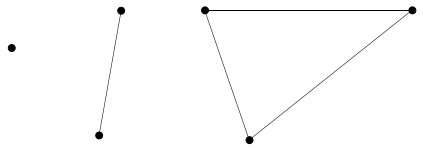

Figure 4.13: *Un ensemble de 4 segments ouverts et 6 points.*

Nous définissons le diagramme de Voronoï de S en faisant l'usage de l'amplitude d'un point et des disques de Delaunay. Soit p un point du plan. L'*amplitude* de p (relativement à S) est le nombre

$$\delta(p) = \min\{d(p,s) \mid s \in S\}.$$

Lorsque s est un segment, la distance de p à s est définie par

$$d(p,s) = \inf\{d(p,q) \mid q \in s\}.$$

Le *disque de Delaunay* du point p du plan est le disque $D(p)$ de centre p et de rayon $\delta(p)$. C'est le plus grand disque centré sur p et dont l'intérieur est disjoint de S. La frontière du disque, qui est le *cercle de Delaunay* de p, rencontre la réunion des objets de S en au moins un point. L'ensemble $Proche(p)$ des primitives *proches* de p est défini comme suit :

- un point $a \in S$ appartient à $Proche(p)$ si $d(a,p) = \delta(p)$, donc si a est sur le cercle de Delaunay de p;
- un segment ouvert s appartient à $Proche(p)$ si $d(a,s) = \delta(p)$, et si de plus la projection orthogonale de p sur la droite contenant s appartient à $\bar{s}$.

Le *diagramme de Voronoï* de S est l'ensemble $\mathrm{Vor}(S)$ formé des points p tels que $Proche(p)$ contient au moins deux primitives. Un *sommet* est un point p tel que $Proche(p)$ contient au moins trois primitives. Une *arête* est un ensemble connexe maximal de points tels que $Proche$ contient exactement deux primitives. Une arête peut être infinie des deux côtés, ou d'un seul côté, ou finie; ses extrémités, lorsqu'elles existent, sont des sommets du diagramme.

La forme et la structure du diagramme de Voronoï de segments sont plus complexes que pour un ensemble de points. Dans le cas du diagramme de Voronoï d'un ensemble de points, les arêtes sont des segments de droites, des demi-droites ou des droites. Il n'en est plus ainsi pour le diagramme de Voronoï de segments : les arêtes peuvent être également des arcs de parabole. Plus précisément, soit s un segment ouvert, d'extrémités a et b. La *bande délimitée par* s, notée $B(s)$, est

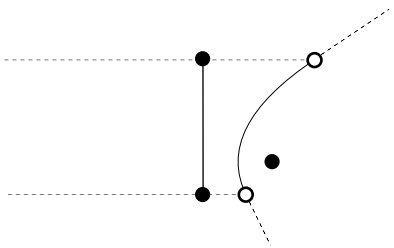

Figure 4.14: *Diagramme de Voronoï formé de 4 demi-droites et d'un arc de parabole.*

la région fermée du plan délimitée par les droites perpendiculaires à s et passant par a et par b respectivement. La bande de s divise le plan en trois régions : deux demi-plans ouverts $P(a)$ et $P(b)$, l'un ayant sur sa frontière a et l'autre b, et la bande $B(s)$ (voir figure 4.14). On a alors :

Lemme 4.8. *Soit p un point qui n'appartient pas au segment $\bar{s} = [a, b]$, et soit E l'ensemble des points équidistants de p et de $[a, b]$. L'intersection de E et de $P(a)$ est la partie de la médiatrice de $[p, a]$ contenue dans $P(a)$; l'intersection de E et de $B(s)$ est la partie de la parabole de foyer p et de directrice $d(a, b)$ contenue dans $B(s)$.* ∎

Ce lemme décrit donc entièrement le diagramme de Voronoï d'un ensemble formé d'un segment et d'un point. Ce diagramme est formé de quatre demi-droites (en pointillé sur la figure), et d'un arc de parabole.

La figure 4.15 donne le diagramme de Voronoï de deux segments. Il a six sommets, et onze arêtes. Nous avons étiqueté certaines arêtes par le couple de primitives qui forme l'ensemble *Proche*. Evidemment, deux arêtes concourantes ont une primitive commune. On peut voir que la partie «centrale» est formée de 7 arêtes.

La *région de Voronoï* d'une primitive r est l'ensemble $R(r)$ des points p tels que r appartient à $Proche(p)$. L'intérieur d'une région de Voronoï est constitué des points p du plan tels que $Proche(p)$ est réduit à une seule primitive. Par connexité, cette primitive est la même pour tous les points de la région. Ainsi, la région $R(s)$ du segment s sur la figure 4.15 est formée de la bande $B(s)$, à laquelle on a enlevé la portion située au-dessus des arêtes (s, d), (s, t), (s, c), (s, t).

Proposition 4.9. *Toute région de Voronoï $R(r)$ est étoilée par rapport à r, c'est-à-dire pour tout $p \in R(r)$, il existe $q \in r$ tel que $[p, q] \subset R(r)$.*

Nous allons prouver un résultat légèrement plus fort :

Proposition 4.10. *Soit p un point à l'intérieur d'une région de Voronoï $R(r)$, et soit $\bar{p}$ la projection de p sur r. Si $p \neq \bar{p}$, l'intersection de la droite D passant par p et $\bar{p}$ et de $R(r)$ est convexe.*

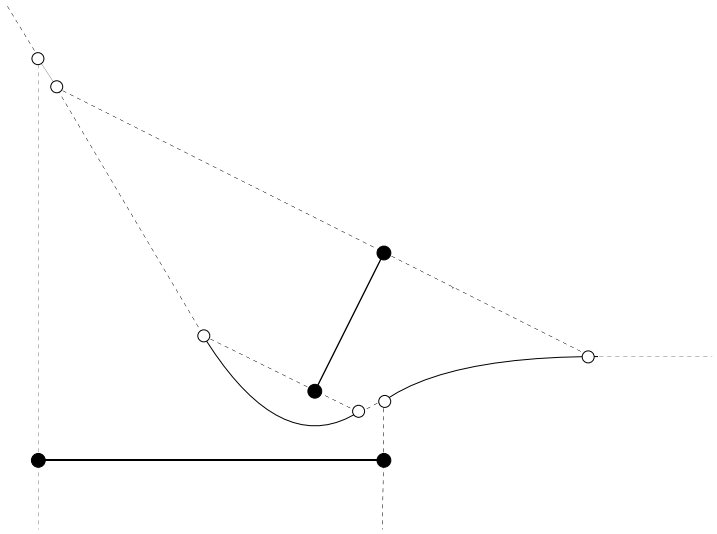

Figure 4.15: *Le diagramme défini par deux segments.*

Preuve. Comme p appartient à $R(r)$, l'intersection du cercle de Delaunay de centre p avec la réunion des primitives ne rencontre que r, et l'intersection est réduite au point $\bar{p}$. Soit q un point de $D \cap R(r)$, et t un point de $[\bar{p}, q]$. Le disque de Delaunay de t est contenu dans le disque de Delaunay de q, et est tangent à ce disque en $\bar{p}$, donc $t \in R(r)$, car l'intérieur du disque de Delaunay de q, et donc de t ne rencontre aucune primitive. Cela implique que l'intersection de D et de $R(r)$ est convexe. ∎

Bien entendu, une région de Voronoï n'est en général pas convexe, dans le cas de segments. La figure 4.16 donne un diagramme de Voronoï plus complexe. Lorsque les segments forment des polygones disjoints, on peut distinguer la partie *intérieure* et la partie *extérieure* du diagramme. Dans la figure 4.16, seule est tracée la partie du diagramme qui est intérieure au rectangle, et extérieure aux autres polygones. Cette configuration est celle qui est rencontrée dans le cadre de la planification de trajectoires.

Nous mentionnons sans preuve le résultat suivant :

Théorème 4.11. *Il existe un algorithme pour calculer le diagramme de Voronoï de n primitives en temps $O(n \log n)$.*

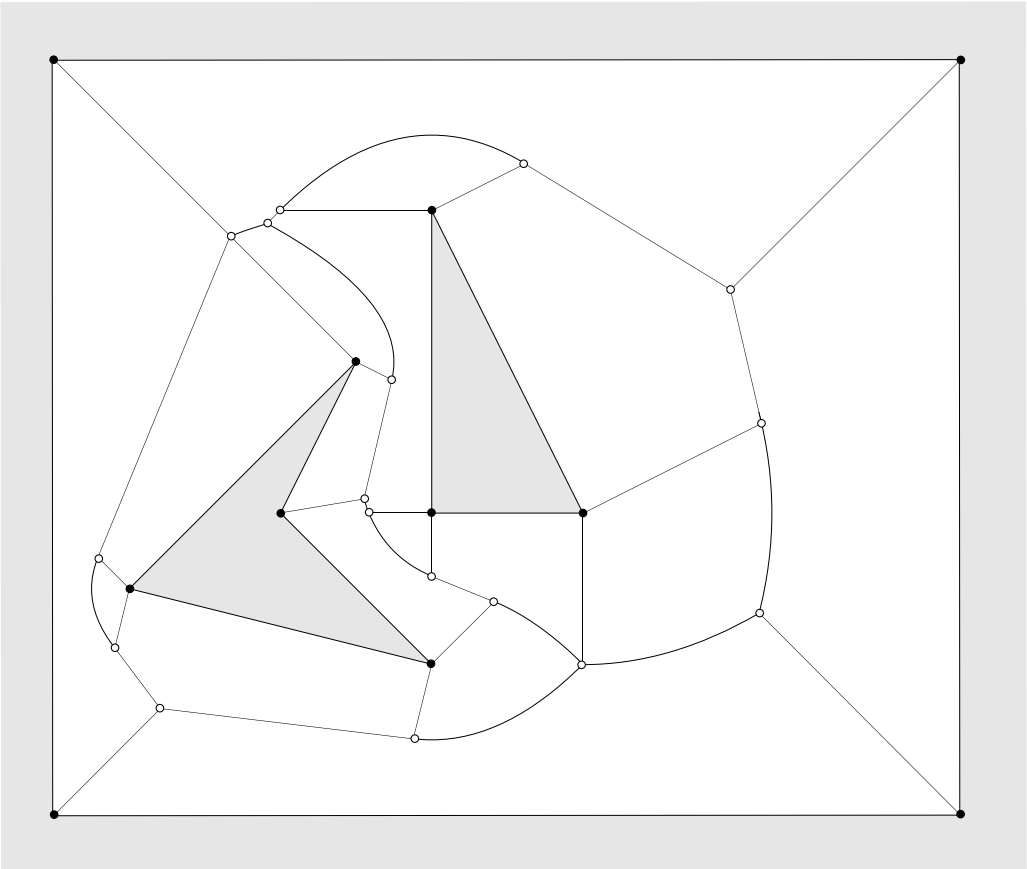

Figure 4.16: *Un diagramme de Voronoï.*

Notes

La littérature sur la géométrie algorithmique est relativement récente, pour les raisons évoquées dans la première section. Les algorithmes présentés dans ce chapitre sont pour la plupart exposés dans les ouvrages de F.P. Preparata et M.I. Shamos et de K. Melhorn :

F.P. Preparata, M.I. Shamos, *Computational Geometry*, Springer, Berlin, 1985.

K. Mehlhorn, *Data Structures and Algorithms : Vol.3 Multi-dimensional Searching and Computational Geometry*, Springer, Berlin, 1984.

Nous avons eu pour principe de donner une description détaillée de l'implémentation des algorithmes proposés, qui garantit la complexité annoncée et qui permet une programmation facile, avec la préoccupation implicite de minimiser les erreurs dues à la manipulation de nombres réels (nous n'avons pas du tout abordé les difficultés posées par la nature non discrète des problèmes géométriques).

Les algorithmes de calcul d'enveloppe convexe dans le plan sont nombreux; le premier algorithme en temps $O(n \log n)$ est dû à Graham et date de 1972.

L'algorithme de la localisation d'un point dans une subdivision planaire exposé est le plus performant à ce jour, en temps et en espace. Il est dû à N. Sarnak et R.E. Tarjan :

N. Sarnak, R.E. Tarjan, Planar point location using persistent search trees, *Comm. Assoc.Comput.Mach.* **29** (1986), 669–679.

L'étude des diagrammes de Voronoï est riche et variée. L'algorithme de calcul du diagramme de Voronoï d'un ensemble fini de points coplanaires exposé ici est celui de S.J. Fortune

S.J. Fortune, A sweepline algorithm for Voronoï diagrams, *Proc. 2nd Ann. ACM Symp. Comput. Geom.* (1986), 313–322.

D'autres algorithmes de même complexité existent, en particulier celui de D.G. Kirkpatrick qui procède par dichotomie. L'avantage de l'algorithme de Fortune est de pouvoir s'adapter au cas des segments ; nous ne l'avons pas présenté ici, car il est relativement sophistiqué.

Le chapitre 7 Computational Geometry de F.F. Yao du *Handbook Vol.A* contient des références complètes sur les sujets abordés dans ce chapitre.

Exercices

11.1. Soit D une droite, $O \in D$, et H un demi-plan ouvert de frontière D. Montrer que la restriction à H de la relation $\preceq_O$ est une relation d'ordre.

11.2. Soit O un point du plan. On insère dans un arbre binaire initialement vide, successivement n points, le critère de descente au sommet contenant p, lors de l'insertion du point q, étant :

si $q \preceq_O p$ alors descendre à gauche sinon descendre à droite.

a) Montrer que le parcours préfixe de l'arbre obtenu fournit un circuit polaire des n points relativement à O.

b) En est-il de même si l'on maintient une structure d'arbre AVL, i.e. si l'on procède à des rééquilibrages par rotations lors des insertions ?

c) Montrer qu'on peut définir une structure d'arbre AVL balisé, i.e. avec stockage des éléments aux feuilles, de façon à ce que l'insertion successive de n points dans un arbre initialement vide fournisse un circuit polaire des n points relativement à O ; on précisera quelles sont les balises aux nœuds et quel est le test de descente.

11.3. Dans l'algorithme de Graham, au lieu de choisir $O(0, y_0)$ à l'intérieur de l'enveloppe convexe, on choisit le « point » $O'(x_0, -\infty)$.

a) Que devient le circuit polaire des n points relativement à O' ?

b) Montrer qu'en appliquant l'algorithme de Graham avec O' au lieu de O, on obtient l'enveloppe convexe supérieure des n points.

c) En déduire un nouvel algorithme de calcul d'enveloppe convexe et donner sa complexité.

11.4. Montrer que dans une gestion dynamique d'enveloppe convexe, on peut réaliser la suppression d'un point en temps $O(\log^2 n)$, où n est le nombre de points considérés.

11.5. On dit que $P = (p_1, p_2, ..., p_n)$ est *une ligne polygonale cadrée dans la direction* δ si c'est une ligne polygonale simple et si de plus, il existe deux droites parallèles distinctes Δ et Δ' de direction δ passant respectivement par p_1 et p_n telles que tous les points p_i $(1 < i < n)$ soient situés à l'intérieur de la bande limitée par Δ et Δ' (figure 4.17(a)).

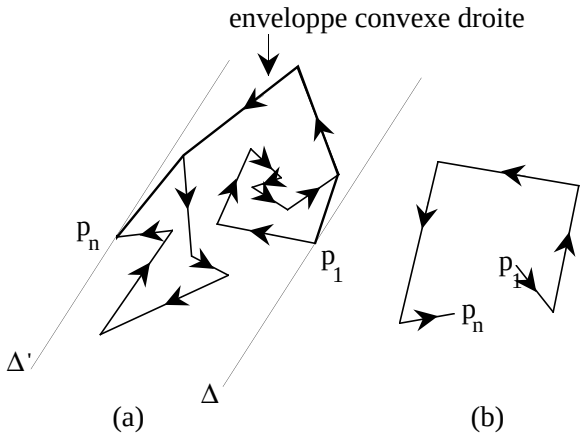

Figure 4.17: *Exemples.*

Il est clair qu'une ligne polygonale simple n'est pas nécessairement cadrée (figure 4.17(b)).

Si $P = (p_1, ..., p_n)$ est une ligne polygonale cadrée, alors p_1 et p_n sont des sommets de l'enveloppe convexe de P, et ils partitionnent le contour orienté de $EC(P)$ en deux lignes polygonales simples, l'une de p_1 vers p_n que nous appellerons *enveloppe convexe droite de* P et l'autre de p_n vers p_1 que nous appellerons *enveloppe convexe gauche*. Cette terminologie se justifie par le fait que l'enveloppe convexe gauche (resp. droite) de P ainsi définie est également le contour de l'enveloppe convexe de l'ensemble des sommets de P situés à gauche (resp. à droite), au sens large, du vecteur $\overrightarrow{p_1 p_n}$. Il suffit donc de calculer l'enveloppe convexe droite D de P, le calcul de l'enveloppe convexe gauche étant de même nature. Ainsi, D est constitué d'une sous-suite $(q_1, ..., q_R)$ de la suite P avec $q_1 = p_1$ et $q_R = p_n$. On peut décrire chaque segment $[q_i, q_{i+1}]$ comme étant la «fermeture» d'une «poche» $K_i = (p_{k_i}, ..., p_{k_{i+1}})$ de la ligne polygonale P (figure 4.18)

a) Déterminer un algorithme qui consiste à suivre la ligne polygonale $(p_1, .., p_n)$ et à construire successivement les fermetures des poches en temps $O(n)$.

b) En déduire un algorithme linéaire de calcul de l'enveloppe convexe d'un polygone simple.

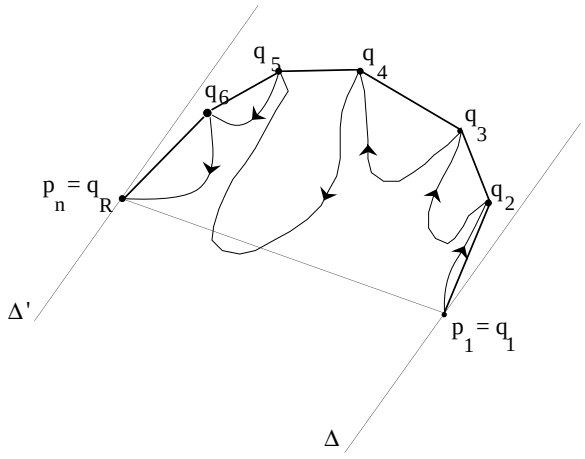

Figure 4.18: *Principe de l'algorithme.*

11.6. Soient P et Q deux polgones convexes ayant respectivement m et n sommets. Montrer qu'on peut calculer la distance de P à Q en temps $O(sup(m,n))$ dans le pire des cas.

11.7. Le problème *des bureaux de poste* :

Etant donnés n points distincts du plan (les bureaux de poste), déterminer pour un point arbitraire p le (ou un) bureau de poste le plus près de p.

Décrire un prétraitement des n points qui permet de résoudre le problème en temps $O(\log n)$.

11.8. Le problème du *plus grand cercle vide* : Etant donnés un ensemble S de n points distincts du plan, déterminer un cercle de plus grand rayon ne contenant aucun des n points à l'intérieur. Montrer que le centre d'un tel cercle est soit un sommet de $Vor(S)$ soit l'intersection d'une arête de $Vor(S)$ et d'un côté de $EC(S)$. En déduire un algorithme en temps $O(n \log n)$ pour résoudre ce problème (algorithme dû à G.T. Toussaint).